OIPE JC50
JAN 2 5 2005
PATENT & TRADEMARK OFFICE

PTO/SB/21 (09-04)

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	10/645,813
Filing Date	August 20, 2003
First Named Inventor	NAKATANI, Yoji
Art Unit	2142
Examiner Name	Jack B. Harvey
Attorney Docket Number	16869S-090800US
Total Number of Pages in This Submission	17

ENCLOSURES *(Check all that apply)*

- [✓] Fee Transmittal Form
  - [ ] Fee Attached
- [ ] Amendment/Reply
  - [ ] After Final
  - [ ] Affidavits/declaration(s)
- [ ] Extension of Time Request
- [ ] Express Abandonment Request
- [ ] Information Disclosure Statement
- [ ] Certified Copy of Priority Document(s)
- [ ] Reply to Missing Parts/ Incomplete Application
  - [ ] Reply to Missing Parts under 37 CFR 1.52 or 1.53
- [ ] Drawing(s)
- [ ] Licensing-related Papers
- [✓] Petition to Make Special
- [ ] Petition to Convert to a Provisional Application
- [ ] Power of Attorney, Revocation Change of Correspondence Address
- [ ] Terminal Disclaimer
- [ ] Request for Refund
- [ ] CD, Number of CD(s) ____________
  - [ ] Landscape Table on CD
- [ ] After Allowance Communication to TC
- [ ] Appeal Communication to Board of Appeals and Interferences
- [ ] Appeal Communication to TC **(Appeal Notice, Brief, Reply Brief)**
- [ ] Proprietary Information
- [ ] Status Letter
- [✓] Other Enclosure(s) (please identify below):

Return Postcard
Eighteen (18) cited references

Remarks: The Commissioner is authorized to charge any additional fees to Deposit Account 20-1430.

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm Name	Townsend and Townsend and Crew LLP		
Signature	[illegible]		
Printed name	Chun-Pok Leung		
Date	January 25, 2005	Reg. No.	41,405

CERTIFICATE OF TRANSMISSION/MAILING

Express Mail Label: **EV 530884140 US**

I hereby certify that this correspondence is being deposited with the United States Postal Service with "Express Mail Post Office to Address" service under 37 CFR 1.10 on this date **January 25, 2005** and is addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below.

Signature	[illegible]		
Typed or printed name	Joy Salvador	Date	January 25, 2005

60406013 v1

OIPE JC50
JAN 2 5 2005
PATENT & TRADEMARK OFFICE

PTO/SB/17 (12-04)

Effective on 12/08/2004.
Fees pursuant to the Consolidated Appropriations Act, 2005 (H.R. 4818).

FEE TRANSMITTAL For FY 2005

☐ Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT	($) 130.00

Complete if Known	
Application Number	10/645,813
Filing Date	August 20, 2003
First Named Inventor	NAKATANI, Yoji
Examiner Name	Jack B. Harvey
Art Unit	2142
Attorney Docket No.	16869S-090800US

METHOD OF PAYMENT (check all that apply)

☐ Check ☐ Credit Card ☐ Money Order ☐ None ☐ Other (please identify): ____________

☒ Deposit Account Deposit Account Number: 20-1430 Deposit Account Name: Townsend and Townsend and Crew LLP

For the above-identified deposit account, the Director is hereby authorized to: (check all that apply)

☒ Charge fee(s) indicated below ☐ Charge fee(s) indicated below, **except for the filing fee**

☒ Charge any additional fee(s) or underpayments of fee(s) under 37 CFR 1.16 and 1.17 ☒ Credit any overpayments

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038

FEE CALCULATION

1. BASIC FILING, SEARCH, AND EXAMINATION FEES

	FILING FEES		SEARCH FEES		EXAMINATION FEES		
		Small Entity		Small Entity		Small Entity	
Application Type	Fee ($)	Fee ($)	Fee ($)	Fee ($)	Fee ($)	Fee ($)	Fees Paid ($)
Utility	300	150	500	250	200	100	
Design	200	100	100	50	130	65	
Plant	200	100	300	150	160	80	
Reissue	300	150	500	250	600	300	
Provisional	200	100	0	0	0	0	

2. EXCESS CLAIM FEES

		Small Entity
Fee Description	Fee ($)	Fee ($)
Each claim over 20 or, for Reissues, each claim over 20 and more than in the original patent	50	25
Each independent claim over 3 or, for Reissues, each independent claim more than in the original patent	200	100
Multiple dependent claims	360	180

Total Claims ______ -20 or HP = ______ Extra Claims x ______ Fee ($) = ______ Fee Paid ($)

HP = highest number of total claims paid for, if greater than 20

Indep. Claims ______ -3 or HP = ______ Extra Claims x ______ Fee ($) = ______ Fee Paid ($)

HP = highest number of independent claims paid for, if greater than 3

Multiple Dependent Claims: Fee ($) ______ Fee Paid ($) ______

3. APPLICATION SIZE FEE

If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).

Total Sheets ______ - 100 = ______ Extra Sheets / 50 = ______ Number of each additional 50 or fraction thereof (round up to a whole number) x ______ Fee ($) = ______ Fee Paid ($)

4. OTHER FEE(S) Fees Paid ($)

Non-English Specification, $130 fee (no small entity discount) ______

Other: **Petitions to the Commissioner** **130.00**

SUBMITTED BY

Signature	[illegible]	Registration No. (Attorney/Agent) 41,405	Telephone 650-326-2400
Name (Print/Type)	Chun-Pok Leung		Date January 25, 2005

60405999 v1

OIPE JC50
JAN 2 5 2005
PATENT & TRADEMARK OFFICE

01-27-05

IFW ~~#~~

PATENT
Attorney Docket No.: 16869S-090800US
Client Ref. No.: W1109-01ES

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:

YOJI NAKATANI et al.

Application No.: 10/645,813

Filed: August 20, 2003

For: METHOD FOR ACCESSING DISTRIBUTED FILE SYSTEM

Customer No.: 20350

Examiner: Jack B. Harvey

Technology Center/Art Unit: 2142

Confirmation No.: 8842

PETITION TO MAKE SPECIAL FOR NEW APPLICATION UNDER M.P.E.P. § 708.02, VIII & 37 C.F.R. § 1.102(d)

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This is a petition to make special the above-identified application under MPEP § 708.02, VIII & 37 C.F.R. § 1.102(d). The application has not received any examination by an Examiner.

(a) The Commissioner is authorized to charge the petition fee of $130 under 37 C.F.R. § 1.17(i) and any other fees associated with this paper to Deposit Account 20-1430.

```
02/01/2005 CNGUYEN  00000012 201430    10645813
01 FC:1464        130.00 DA
```

(b) All the claims are believed to be directed to a single invention. If the Office determines that all the claims presented are not obviously directed to a single invention, then Applicants will make an election without traverse as a prerequisite to the grant of special status.

(c) Pre-examination searches were made of U.S. issued patents, including a classification search, a computer database search, and a keyword search. The searches were performed on or around July 15, 2004, and were conducted by a professional search firm, Kramer & Amado, P.C. The classification search covered Class 709 (subclass 217), Class 710 (subclasses 8, 62, and 203), and Class 711 (subclasses 154 and 202) for the U.S. and foreign subclasses identified above. The computer database search was conducted on the USPTO systems EAST and WEST. The keyword search was conducted in Class 709 (subclasses 203, 219, and 229). The inventors further provided two references considered most closely related to the subject matter of the present application (see references #5-6 below), which were cited in the Information Disclosure Statements filed on August 20, 2003.

(d) The following references, copies of which are attached herewith, are deemed most closely related to the subject matter encompassed by the claims:

(1) U.S. Patent Publication No. 2002/0112023 A1;

(2) European Patent Publication No. EP 0794479 A1;

(3) U.S. Patent Publication No. 2002/0083120 A1;

(4) U.S. Patent Publication No. 2002/0161860 A1;

(5) U.S. Patent Publication No. 2002/0152339 A1;

(6) U.S. Patent No. 6,446,141 B1;

(7) U.S. Patent No. 5,012,405;

(8) U.S. Patent No. 5,689,701;

(9) U.S. Patent No. 5,752,060;

(10) U.S. Patent No. 5,761,498;

(11) U.S. Patent No. 6,006,018;

(12) U.S. Patent No. 6,026,414;

(13) U.S. Patent No. 6,606,690 B2;

(14) U.S. Patent No. 6,718,372 B1;

(15) U.S. Patent Publication No. 2003/0101200 A1;

(16) U.S. Patent Publication No. 2003/0135514 A1;

(17) U.S. Patent Publication No. 2004/0019655 A1; and

(18) Japanese Patent Publication No. JP 2003-162441.

(e) Set forth below is a detailed discussion of references which points out with particularity how the claimed subject matter is distinguishable over the references.

A. Claimed Embodiments of the Present Invention

The claimed embodiments relate to a distributed file system (DFS) that allows access to a DFS file using a conventional protocol without making a modification on a side of a client that uses the conventional protocol.

Independent claim 24 recites a gateway apparatus between a client computer requesting a file access and a file server executing file access processes according to a received file access request from the client computer. The gateway apparatus comprises a first type protocol processing unit which is configured to receive a first type file access request according to the first type protocol from the client computer and respond to the received first type file access request, wherein a first type file system according to the first type protocol is a directory structural file system and the first type file access request includes a path ID indicating a directory including a target file and a first type file ID indicating the target file, and the first type file ID is a unique ID in the directory. The gateway apparatus further comprises a second type file system access unit which is configured to receive a file access request from the first type protocol processing unit and issue a second type file access request to a second type file system, wherein the second type file access request includes a second type file ID indicating the target file, and the second type file ID is a unique ID in the second type file system and assigned to the target file by the second type file system. A directory management unit is configured to manage a correspondence between a directory structure of the first type file system and a second type file ID of the second type file system, wherein the second type file ID used by the second type file system access unit is specified by the directory management unit based on the first type file ID included in the first type file access request.

Independent claim 40 recites a computer readable storage medium having a computer program for a gateway apparatus to manage file access between a client computer requesting a file access and a file server executing file access processes according to a received file access request from the client computer. The computer program comprises code for a first type protocol processing unit to receive a first type file access request according to the first type protocol from the client computer and respond to the received first type file access request, wherein a first type file system according to the first type protocol is a directory structural file system and the first type file access request includes a path ID indicating a directory including a target file and a first type file ID indicating the target file, and the first type file ID is a unique ID in the directory. The computer program further comprises code for a second type file system access unit to receive a file access request from the first type protocol processing unit and issue a second type file access request to a second type file system, wherein the second type file access request includes a second type file ID indicating the target file, and the second type file ID is a unique ID in the second type file system and assigned to the target file by the second type file system. The computer program also comprises code for a directory management unit to manage a correspondence between a directory structure of the first type file system and a second type file ID of the second type file system, wherein the second type file ID used by the second type file system access unit is specified by the directory management unit based on the first type file ID included in the first type file access request.

One of the benefits that may be derived is that access to a file on the file system using file ID can be made for the directory structural file system, without making a modification on the client's side.

B. Discussion of the References

None of the following references disclose providing a gateway function between two file systems by managing a directory structure of a directory structural file system and a file ID assigned by a file system and unique in the file system. For instance, the references do not teach a directory management unit configured to manage a correspondence between a directory structure of the first type file system and a second type file ID of the second type file system, wherein the second type file ID used by the second type file system access unit is specified by the directory management unit based on the first type file ID included in the first type file access request. The references merely disclose distributed file systems.

1. U.S. Patent Publication No. 2002/0112023 A1

This reference relates to a system and a method for providing a plurality of client applications access to data in a distributed file system. Read requests are separated from write requests, so that read requests are processed by dedicated read servers and write requests are processed by a dedicated write server. A DFS server (184) receives file access requests from the client application (156). The DFS server is implemented with conventional server software for a distributed file system, for example, NFS server software. If the DFS client interface (158) or load balancer (164) sends only read requests to the DFS server, the DFS server processes only read requests and commercially available DFS server software is used. See paragraphs [0023]-[0033]. The reference fails to teach providing a gateway function between two file systems by managing a directory structure of a directory structural file system and a file ID assigned by a file system and unique in the file system.

2. European Patent Publication No. EP 0794479 A1

This reference discloses a method and an apparatus for providing dynamic distributed file system client authentication. One method for providing dynamic distributed file system client authentication within a distributed file system computing environment includes receiving an NFS (Network File System) request from an NFS client, determining whether the NFS client has an access status sufficient to perform the NFS request, and performing the NFS request when the NFS client has sufficient access status. The dynamic NFS client authentication service 270 considers factors such as time, date, identity of the NFS client, a nature of the NFS request, and a current status of a resource upon which the NFS request operates. See column 8, lines 4-25; column 10, lines 9-31. The reference does not teach providing a gateway function between two file systems by managing a directory structure of a directory structural file system and a file ID assigned by a file system and unique in the file system.

3. U.S. Patent Publication No. 2002/0083120 A1

This reference discloses a shared storage distributed file system providing applications with transparent access to a storage area network (SAN) attached storage device. This is accomplished by providing clients read access to the devices over the SAN and by requiring most write activity to be serialized through a network attached storage (NAS) server. See paragraphs [0089]-[0090]. A supplemental read path is provided through the NAS server for those circumstances where the local file system is unable to provide valid data reads. See paragraphs [0093]-[0095]. The reference fails to teach providing a gateway function between two file systems by managing a directory structure of a directory structural file system and a file ID assigned by a file system and unique in the file system.

4. U.S. Patent Publication No. 2002/0161860 A1

This reference relates to a method and a system providing a distributed file system and distributed file system protocol utilizing a version-controlled file system with two-way differential transfer across a network. Fig. 4A shows a DDFS (Differential Distributed File System) filesystem protocol by way of several examples of a two-way differential file transfer protocol that refers to the remote clients. See paragraphs [0135]-[0146]. Fig. 4B illustrates how the client processes a DDFS file read request in which the client receives a delta ("diff") between the latest version of the file and the version that the client has cached. See paragraphs [0147]-[0148]. Fig. 4C illustrates how the client processes a write request in which the client calculates a delta from the new version of the file and the most recently saved version from the cache. See paragraphs [0149]-[0152].

This reference merely discloses a differential distributed file system. It does not teach providing a gateway function between two file systems by managing a directory structure of a directory structural file system and a file ID assigned by a file system and unique in the file system.

5. U.S. Patent Publication No. 2002/0152339 A1

This reference relates to a storage system that includes a storage controller 14 and storage media for reading data from or writing data to the storage media in response to SCSI, NFS, CIFS, or HTTP type read/write requests. The storage controller includes SCSI, NFS, CIFS, and HTTP interface adapters 26, 28, 30, 32 for receiving the read/write requests and effecting the reading of data from or the writing of data to the storage media. See paragraphs [0019]-[0021]. The reference fails to teach providing a gateway function between two file systems by managing a directory structure of a directory structural file system and a file ID assigned by a file system and unique in the file system.

6. U.S. Patent No. 6,446,141 B1

This reference discloses a system equipped with a communication interface for connection to different kinds of user data from a storage server. The storage server comprises a bus system which includes a plurality of slots having interfaces to respective data stores; and an operating system which includes logic controlling transfers among the plurality of slots over the bus system according to an internal format, logic for translating a storage transaction received over the communication interface into the internal format, logic for configuring the plurality of slots according to a configuration data, and logic to monitor the performance and condition of the storage server. See column 1, line 55 to column 2, line 10.

This reference merely discloses a communication interface for connection to different kinds of user data from a storage server. It does not teach providing a gateway function between two file systems by managing a directory structure of a directory structural file system and a file ID assigned by a file system and unique in the file system.

7. U.S. Patent No. 5,012,405

This reference discloses a file management system for permitting user access to files in a distributed file system based on linkage relation information. The information processing devices are linked through a communication line and dispersed files are utilized by a plurality of users in common. Each information processing device 4a-4c is provide with a file system management program 9a-9c, user programs 12x-12z, 12b, 12c, and a management table 10, 11b, 11c (col. 6, lines 45-49; Fig. 3).

This reference merely discloses a file management system for a distributed file system based on linkage relation information. It does not teach providing a gateway function between two file systems by managing a directory structure of a directory structural file system and a file ID assigned by a file system and unique in the file system.

8. U.S. Patent No. 5,689,701

This reference discloses a system and a method for providing compatibility between distributed file system namespaces and operating system pathname syntax. A DFS pathname prefix is associated with each drive letter that is attached to a DFS IFS driver. Before an IFS driver is used, an application program issues a command to associate a drive letter with a particular IFS driver. The command issued also carries a DFS pathname prefix within a data buffer. The IFS services the command by validating existence of the DFS pathname prefix, and thereafter stores such prefix into an internal table of the buffer where it is associated with the attached drive letter. File system requests later received by the DFS client IFS driver carrying a pathname containing that drive letter will have their file specifications edited by the DFS code prior to processing. The drive letter in the pathname is replaced by the DFS pathname prefix from the IFS driver's internal table, and operating system slashes in operating system pathname are converted to DFS slashes. The operating system user may thereby reference DFS objects relative to a point in the DFS namespace using the operating system's pathname syntax with which the user is more comfortable. Fig. 13 shows a flow diagram of a process in which a table is built correlating operating system drive letters and file system namespace syntax. Fig. 14 is a flow diagram of a process for processing a file system request whereby compatibility is provided between distributed file system namespace and operating system pathname syntax not otherwise natively supported in the namespace.

This reference merely discloses providing compatibility between distributed file system namespace and operating system pathname syntax. It does not teach providing a gateway function between two file systems by managing a directory structure of a directory structural file system and a file ID assigned by a file system and unique in the file system.

9. U.S. Patent No. 5,752,060

This reference discloses a file access scheme in a distributed data processing system for executing an access to a file in response to a file server to effect a data processing includes a device for executing an input/output processing such that each processing module unit operates as a subroutine of a data processing unit and executes input/output processing to the file controlled by itself and a device for executing data transfer processing such that the processing module unit operates as a subroutine of the data processing unit and implements a data transfer processing between the processing module unit and the file server of other processing modules. As shown in Fig. 2, when an open instruction of a file 11-i is made to a file server 13-i in a processing module unit 1-i which is part of a distributed data processing system, and the data processing unit 10-j which has submitted the open instruction belongs to the same processing module unit to which it belongs 1-i, that is, i=j, the activation of an input/output processing unit 14-i ready for the data processing unit 10-i as a subroutine is designated. The data processing unit 10-i which has submitted the open instruction employs its self data area and executes an input/output request for the disk 12-i to effect an access processing to the file 11-i.

This reference merely discloses a file access scheme in a distributed data processing system. It does not teach providing a gateway function between two file systems by managing a directory structure of a directory structural file system and a file ID assigned by a file system and unique in the file system.

10. U.S. Patent No. 5,761,498

This reference discloses a distribution file system for accessing required portion of a file. When a host system 13 opens a file, it can see it as a file. When each system 16 opens a file, it can see the file as a local file. A substance is stored as one file in a secondary memory. In an open mode, however, each cell can see a required portion corresponding to the distribution information stored in the distribution information storing portion 15. See Figs. 2A-2C; column 3, line 64 to column 4, line 6.

This reference merely discloses a particular implementation of a distribution file system. It does not teach providing a gateway function between two file systems by managing a directory structure of a directory structural file system and a file ID assigned by a file system and unique in the file system.

11. U.S. Patent No. 6,006,018

This reference discloses a translation gateway for a distributed computing environment including a source computer system and a target computer system, each of which has at least one client, one server and a distributed file system. A method for providing authenticated access to files stored in the target distributed file system in response to file requests originating from clients associated with the source distributed file system begins by mapping credentials associated with incoming client requests from the source distributed file system into enhanced credentials containing authentication information associated with an authentication model of the target distributed file system (see Fig. 6; col. 9, lines 36-55). At least one enhanced credential is then augmented with one or more attributes whose values may be extracted and used in the processing of the file system request by the target file system (see Fig. 5; col. 9, lines 14-35). The source computer system's server then makes file system requests using the enhanced credentials so that each file request appears to the target computer client as if it were made by an authenticated process with equivalent attributes (see Fig. 4; col. 8, lines 13-57).

This reference merely discloses a distributed file system translator with extended attribute support. It does not teach providing a gateway function between two file systems by managing a directory structure of a directory structural file system and a file ID assigned by a file system and unique in the file system.

12. U.S. Patent No. 6,026,414

This reference discloses a system for backing up files in a distributed computing system. A file server 8 maintains files in a shared name space. The client 4 and proxy client 10 include a distributed file system (DFS) client program 14 that provides communication with the file server 8 and access to files in the shared name space. The client 4 and the proxy client 10 each have a backup client program 18. The file server 8 provides the first backup client program and the second backup client program with access to the files in the shared name space. The first backup client program initiates a backup request to backup a requested file. A determination is made as to whether the requested file is maintained in a shared name space. The backup request is transmitted to the second backup client program upon determining that the requested file is maintained in the shared name space. The second backup client program transmits a message to the file server to provide the requested file. The file server transmits the requested file with the file server to the second backup client program. The second backup client program then transmits the requested file to a backup server program 20. The backup server program stores the requested file in a storage device.

This reference merely discloses a system for backing up files in a distributed computer system. It does not teach providing a gateway function between two file systems by managing a directory structure of a directory structural file system and a file ID assigned by a file system and unique in the file system.

13. U.S. Patent No. 6,606,690 B2

This reference discloses a system for accessing a plurality of storage devices in a storage area network (SAN) as network attached storage (NAS) in a data communication network is described. A SAN server includes a first interface and a second interface. The first interface is configured to be coupled to the SAN. The second interface is coupled to a first data communication network. A NAS server includes a third interface and a fourth interface. The third interface is configured to be coupled to a second data communication network. The fourth interface is coupled to the first data communication network. The SAN server allocates a first portion of the plurality of storage devices in the SAN to be accessible through the second interface to at least one first host coupled to the first data communication network. The SAN server allocates a second portion of the plurality of storage devices in the SAN to the NAS server. The NAS server configures access to the second portion of the plurality of storage devices to at least one second host coupled to the second data communication network. See column 2, lines 40-67.

This reference merely discloses a system for accessing a plurality of storage devices in a SAN as NAS in a data communication network. It does not teach providing a gateway function between two file systems by managing a directory structure of a directory structural file system and a file ID assigned by a file system and unique in the file system.

14. U.S. Patent No. 6,718,372 B1

This reference discloses a data server that can provide access to data, such as mainframe data, by open systems platforms. The system includes a shared storage interface 214 coupling a first computing system 201-1 to a shared storage device 211 in which the shared data is maintained by a second computing system 202 in a manner that is not natively compatible to the first computing system. The computing system further includes a data access server 210 which executes on the processor in the first computing system. When executing, the data access server receives, via the network interface 213, a client message to access data on the shared storage device and in response to receiving the client message, retrieves, via the network interface, data storage information provided from the second computing system coupled to the first computing system. The data storage information is stored in the memory system and allows the data access server on the first computing system to access the data in the shared storage device in a manner that is compatible with the first computing system. See Figs. 5-6.

This reference merely discloses a data access server for open systems platforms. It does not teach providing a gateway function between two file systems by managing a directory structure of a directory structural file system and a file ID assigned by a file system and unique in the file system.

15. U.S. Patent Publication No. 2003/0101200 A1

This reference discloses a distributed file sharing system and a file access control method of efficiently searching for access rights. With a shared index information file 1244 in a file sharing index manager 120, 140, 160, 180, a distributed file system controls access to files based on access right obtained from the index information. Even when a host terminal 12, 14, 16, 18, 20 operated by a user does not have directory information required, that host terminal may obtain an access right from the file sharing index manager without making access to the host terminals. The host terminals perform local management via file sharing managers to minimize accesses to the host terminals which are required until processing is completed. See Fig. 1; and paragraphs [0034]-[0037].

This reference merely discloses a particular control method of searching for access rights in a distributed file sharing system. It does not teach providing a gateway function between two file systems by managing a directory structure of a directory structural file system and a file ID assigned by a file system and unique in the file system.

16. U.S. Patent Publication No. 2003/0135514 A1

This reference discloses a distributed file system incorporating a virtual hot spare. The intelligent distributed file system 110 enables the storing of file data among a plurality of smart storage units 114 which are accessed as a single file system. The intelligent distributed file system utilizes a metadata data structure to track and manage detailed information about each file, including, for example, the device and block locations of the file's data blocks, to permit different levels of replication and/or redundancy within a single file system, to facilitate the change of redundancy parameters, to provide high-level protection for metadata, to replicate and move data in real-time, and to permit the creation of virtual hot spares among the smart storage units without the need to idle any single smart storage unit in the intelligent distributed file system. See Fig. 1; and paragraphs [0040]-[0045].

This reference merely discloses a specific implementation of a distributed file system. It does not teach providing a gateway function between two file systems by managing a directory structure of a directory structural file system and a file ID assigned by a file system and unique in the file system.

17. U.S. Patent Publication No. 2004/0019655 A1

This reference discloses a method for forming such a virtual network storage with use of general network storages and through the processings by a network storage accessing protocol without using any of dedicated network storages, concentrated management servers, and distributed directories. The virtual network storage, when receiving a READDIR request from a client in step 301, transfers the received READDIR request to each network storage in step 302, then receives READDIR responses from the network storages. The virtual network storage then combines the READDIR responses from the network storages and sends the result to the client. See paragraphs [0035]-[0037].

This reference merely discloses a method for forming virtual network storage. It does not teach providing a gateway function between two file systems by managing a directory structure of a directory structural file system and a file ID assigned by a file system and unique in the file system.

18. Japanese Patent Publication No. JP 2003-162441

This reference contains the same disclosure as reference item 15 (U.S. Patent Publication No. 2003/0101200 A1).

(f) In view of this petition, the Examiner is respectfully requested to issue a first Office Action at an early date.

Respectfully submitted,

[illegible]

Chun-Pok Leung
Reg. No. 41,405

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, 8th Floor
San Francisco, California 94111-3834
Tel: 650-326-2400
Fax: 415-576-0300
Attachments
RL:rl
60405822 v1

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 794 479 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
10.09.1997 Bulletin 1997/37

(51) Int Cl.6: **G06F 1/00**

(21) Application number: **97300619.0**

(22) Date of filing: **30.01.1997**

(84) Designated Contracting States:
DE FR GB IT SE

(30) Priority: **04.03.1996 US 610704**

(71) Applicant: **SUN MICROSYSTEMS, INC.**
Mountain View, CA 94043 (US)

(72) Inventors:
- **Callaghan, Brent P.**
Mountain View, California 94040 (US)
- **Eisler, Michael R.**
San Jose, California 95135 (US)

(74) Representative: **Browne, Robin Forsythe, Dr.**
Urquhart-Dykes & Lord
Tower House
Merrion Way
Leeds LS2 8PA West Yorkshire (GB)

(54) Method and apparatus for providing dynamic network file system client authentication

(57) A variety of methods and apparatus are taught for providing dynamic distributed file system client authentication. One method for providing dynamic distributed file system client authentication within a distributed file system computing environment includes the steps of receiving an NFS request from an NFS client, determining whether the NFS client has an access status sufficient to perform the NFS request, and performing the NFS request when the NFS client has sufficient access status. In some embodiments, the NFS request includes a file handle representing a given file system available on the server computer system and a file operation to be performed upon the given file system. A server computer in accordance with one embodiment of the present invention is operable to provide dynamic NFS client authentication. The server computer includes a CPU, a RAM accessible by the CPU, a ROM accessible by the CPU, a network I/O port coupled with the CPU, a mass storage device accessible by the CPU, and a kemel implemented on the server computer. In addition, the server computer implements a dynamic NFS client authentication service operable to receive an NFS request from an NFS client and to authenticate the NFS client in relation to the NFS request. The dynamic NFS client authentication service considers factors such as time, date, identity of the NFS client, a nature of the NFS request, and a current status of a resource upon which the NFS request operates.

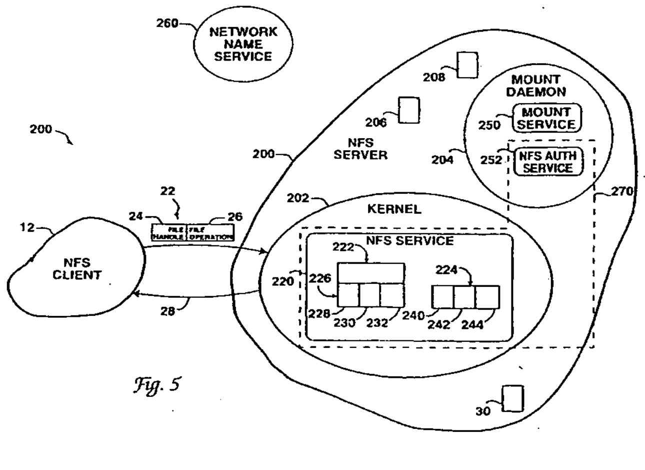


Fig. 5

EP 0 794 479 A1

Description

BACKGROUND OF THE INVENTION

The present invention relates generally to file sharing over a computer network. More specifically, the present invention teaches methods and apparatus for providing dynamic client authentication in a distributed computer file system.

Sun Microsystems, Inc.'s "Distributed File System", designated as NFS®, is a computer implemented service designed to allow computer systems to share files across a computer network. In brief, file systems are mounted across the network, making them appear as if a local computer system is accessing the file system locally when in fact the files are stored on a remote server computer. Thus, using NFS, it is possible to share individual files, file hierarchies, and entire file systems across a network.

NFS employs a client/server paradigm. A computer that wishes to share its file system with other computers on the network acts as a server computer. Files are physically located on and managed by the server computer. A separate computer that wishes to access files located on the server computer acts as a client of the server computer. In order to access files located on the server computer, the client computer first mounts the required file system and then makes file access requests across the network to the server. In general, a computer may simultaneously operate as a client and a server.

Fig. 1 diagramatically illustrates an NFS client/server paradigm 10 of the prior art. The NFS client/server paradigm 10 includes an NFS client 12 and an NFS server 14. The NFS server 14 includes a kernel 16 and a mount daemon 18. As will be well familiar to those of skill in the art, the kernel 16 typically implements the most primitive functions of the server's operating system. Additionally, because the kernel 16 is generally resident in random access memory (RAM), it is sound programming strategy to minimize the memory space required by these primitive functions.

The mount daemon 18 is a process implemented on the server 14 which autonomously answers file system mount requests, making available those file systems which the clients may legitimately access. When the NFS client 12 attempts to mount a given file system 30, the mount daemon 18 authenticates that the NFS client 12 is entitled to access the given file system 30 and, if so, returns a file handle 24 corresponding to the given file system 30. The file handle 24 serves as a key facilitating all further requests between the NFS client 12 and the NFS server 14 with regards to the given file system 30.

Once the NFS client 12 obtains a file handle 24, all file system requests are handled by an NFS service 20 implemented within the kernel 16. Each file system request such as NFS request 22 includes both the file handle 24 and a file operation 26. When the file handle 24 is valid, the NFS service 20 executes the file operation 26 as a matter of course, without authenticating the NFS client 12. When necessary, the NFS service 20 returns an NFS response 28, providing the NFS client 12 with either the requested file information or a message indicating success or failure of the requested file operation 26.

While the prior art NFS paradigm 10 provides resource sharing across a network, it inherently creates a potential for security risks within the network. As used herein, security risks include unauthorized access to resources found on an NFS server computer. In particular, prior art NFS implementations only provide what is herein termed static client authentication mechanisms.

A static client authentication mechanism operates only once with respect to a client's log in session: initially when the client attempts to mount resources. In perhaps the least secure situations the mount daemon 18 simply verifies that the NFS client 12 is entitled to access by comparing the NFS client 12 and the mount request with the client's access status stored in a file generally called sharetab (for share table). As will be appreciated, a client's access status to a given file system 30 can be either "no access", "ro" for read only access, or "rw" for read and write access. When the client's access status satisfies the mount request, the NFS client 12 receives a valid file handle 22 for use in subsequent NFS requests.

Therefore, a static client authentication mechanism can protect NFS servers from unauthorized NFS clients lacking a valid file handle. However, even the more sophisticated static client authentication mechanism relies on the assumption that clients having valid file handles are authorized to access the server's file system corresponding to the valid file handle. No protection is provided against attacking clients who have guessed or misappropriated valid file handles.

Fig. 2 is a flow chart illustrating a security breach 50 of an NFS server 14 by an attacking client 12 having a valid file handle 24. The breach 50 starts in a step 52. At step 52 the attacking client has unauthorized possession of a valid file handle 24. The attacking client may have guessed or misappropriated the valid file handle 24 by eavesdropping on the network. In a step 54 the attacking client 14 makes an NFS request 22 including the valid file handle 24. Because the current NFS 22 request includes the valid file handle 22, the NFS service 20 performs the requested file operation 26. Then, in step 56, the attacking client receives back the desired response and security of the server 14 is breached.

While the example of Fig. 2 focused on security risks posed by attacking clients, security problems exist even with respect to clients whose access status has changed subsequent to mounting the given file system 30. This is because, once an NFS client 12 has mounted within the prior art NFS client/server paradigm 10, the only way an NFS server 14 can enforce the client's new access status to given file system 30 is to force the NFS client 12 to unmount the given file system 30 and then mount the given file system 30 again.

Accordingly, what is needed is a dynamic NFS client authentication mechanism which provides NFS client authentication upon every NFS request. Such a dynamic NFS client authentication mechanism should insure that only authorized clients are allowed to access a server's file systems, regardless of whether the client's request includes a valid file handle. In addition, the dynamic NFS client authentication mechanism should enable a server to dynamically alter a client's access status without altering a client's server connection status.

SUMMARY OF THE INVENTION

To achieve the foregoing and other objectives and in accordance with the purpose of the present invention, a variety of methods and apparatus are disclosed herein. A first aspect of the present invention teaches a method for providing dynamic network file system client authentication within a distributed file system computing environment. The method is implemented upon an NFS server computer system and includes the steps of receiving a network file system request from an NFS client, determining whether the NFS client has an access status sufficient to perform the NFS request, and performing the NFS request when the NFS client has sufficient access status. According to some embodiments, the NFS request includes a file handle representing a given file system available on the server computer system and a file operation to be performed upon the given file system.

In accordance with another aspect, an export information table is resident on the server computer system. An entry in the export information table for the given file system includes a read only bit and a read-write bit. The read only bit indicates global read only access status, while the read-write bit indicates global read and write access status. According to this aspect, the export information table is searched to determine whether the NFS client has an access status sufficient to perform the NFS request. When the read only bit is set, the client's access status is set to read only. Similarly, when the read-write bit is set, the client's access status is set to read-write. Thus when the entry in the export information table is determinative of the client's access status, it is then directly determined from the client's access status whether the requested NFS operation can be performed.

In a further related aspect, when the entry in the export information table is not determinative of the client's access status (neither bit is set), a cache memory is searched for a specific export authentication cache entry for the NFS client which corresponds to the given file system. When present, the specific export authentication cache entry indicates the client's access status for the given file system thereby enabling direct determination of whether the requested NFS operation can be performed. When not present, the specific export authentication cache entry is first created.

One embodiment of the present invention teaches a server computer for use in a NFS computing environment, the server computer operable to provide dynamic NFS client authentication. The server computer includes a CPU, a RAM accessible by the CPU, a ROM accessible by the CPU, a network I/O port coupled with the CPU, a mass storage device accessible by the CPU, and a kernel implemented on the server computer. The mass storage device is capable of storing a given file system modifiable by clients of the server computer having an access status of read-write for the given file system, readable by clients of the server computer having the access status of read only for the given file system, and inaccessible to other clients. In addition, the server computer implements a dynamic NFS client authentication service operable to receive an NFS request from an NFS client and to authenticate the NFS client in relation to the NFS request. The dynamic NFS client authentication service considers factors such as time, date, identity of the NFS client, a nature of the NFS request, and a current status of a resource upon which the NFS request operates.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention, together with further objectives and advantages thereof, may best be understood by reference to the following description taken in conjunction with the accompanying drawings in which:

FIGURE 1 is a pictorial illustration of an NFS client/server paradigm of the prior art;
FIGURE 2 is a flow chart showing a security breach of an NFS server computer by an attacking NFS client computer having a valid file handle;
FIGURE 3 is a pictorial illustration of various computers linked together in a computer network;
FIGURE 4 illustrates diagrammatically the major components of a computer in Fig. 3;
FIGURE 5 is a pictorial illustration of an NFS client/server paradigm in accordance with one embodiment of the present invention;
FIGURE 6 is a flow chart showing a process by which an NFS server starts in accordance with another embodiment of the present invention;
FIGURE 7 is a flow chart showing a method by which an NFS client makes an NFS request for which the NFS client is authorized, the method in accordance with one aspect of the present invention;
FIGURE 8 is a flow chart showing a method by which an NFS client makes an NFS request for which the NFS client is not authorized, the method in accordance with another aspect of the present invention;
FIGURE 9 is a flow chart showing a method by

which an NFS server performs dynamic NFS client authentication with regards to an NFS request in accordance with yet another aspect of the present invention;

FIGURE 10 is a flow chart providing a more detailed showing of step 436 of Fig. 9, the method of Fig. 10 in accordance with a further aspect of the present invention;

FIGURE 11 is a flow chart showing one method for performing that portion of dynamic NFS client authentication which occurs in the NFS server's kernel, the method in accordance with yet another aspect of the present invention;

FIGURE 12 is a flow chart showing a method for performing that portion of dynamic NFS client authentication which occurs external to the NFS server's kernel, the method in accordance with a still further aspect of the present invention; and

FIGURE 13 is a flow chart showing a method for temporarily modifying the access status of an NFS client with respect to a given file system on an NFS server, the method in accordance with one aspect of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

In a preferred embodiment of the present invention, a distributed file system computing environment is implemented on a server computer and one or more client computers linked together by a network. The network may take any suitable form. By way of example, a representative network arrangement 100 is illustrated in Fig. 3. The network arrangement 100 includes a first computer 102 which is coupled to a transmission line 104. The network 100 further includes a router or the like 106 in addition to other computers 108, 110, and 112 such that NFS requests and NFS replies can be passed among the networked computers. As will be appreciated, any of computers 102, 106, 108, 110, and 112 may be configured as an NFS server, an NFS client, or both. The design, construction and implementation of computer networks will be familiar to those of skill in the art.

A representative computer 130 suitable for use as computers 102, 108, 110, and/or 112 of Fig. 3 is illustrated schematically in Fig. 4. Computer 130 includes a central processing unit (CPU) 132 which is coupled with random access memory (RAM) 134 and with read only memory (ROM) 136. Typically, RAM 134 is used as a "scratch pad" memory and includes programming instructions and data for processes currently operating on CPU 132. ROM 136 typically includes basic operating instructions and data used by the computer 130 to perform its functions. In addition, a mass storage device 138, such as a hard disk, CD ROM, magneto-optical (floptical) drive, tape drive or the like, may be optionally coupled with CPU 132.

The mass storage device 138 is optional for an NFS client, but typically an essential element of an NFS server. This is because, in order to play a useful role, the NFS server ought to maintain substantial file systems. However, the methods and apparatus of the present invention may be implemented upon a computer 130 which does not include a mass storage device 138. The mass storage device 138 of an NFS server includes data in the form of file systems potentially accessible by all NFS clients on the network 100. In addition, the mass storage device 138 often includes additional programming instructions, data and objects that typically are not in active use by the CPU 132, although the address space may be accessed by the CPU 132, *e.g.,* for virtual memory or the like.

Each of the above described computers includes a network input/output source 140 which is coupled with a network such as network 100. The network input/output source may take any suitable form. Further, the above described computers optionally includes an additional input/output source 142 such as a keyboard, pointer devices (*e.g.,* a mouse or stylus) and/or display connections. It will be appreciated by those skilled in the art that the above described hardware and software elements, as well as the networking devices, are of standard design and construction, and will be well familiar to those skilled in the art.

Turning next to Fig. 5, an NFS client/server paradigm 200 in accordance with one embodiment of the present invention will now be described. The NFS client/server paradigm 200 includes an NFS client 12 and an NFS server 200. The NFS client 12 and the NFS server 200 may take any suitable form such as a computer 130. The NFS client 12 and the NFS server 200 are typically connected over a network such as network 100 and may communicate via NFS requests and responses such as NFS request 22 and an NFS response 28. In preferred embodiments of the present invention, the NFS request 22 follows a format identical to that of the prior art NFS client/server paradigm 10, having a file handle 24 and a file operation 26. Therefore, preferred embodiments of the present invention are backwards compatible with prior art NFS paradigms. As will be appreciated, the file handle 24 is an identifier or key to a given file system 30 provided to the NFS client 12 during an earlier successful mount request. In general, the given file system 30 may represent any NFS resource available on the server computer. Example NFS resources include such resources as a file and a file system hierarchical structure.

Included in the NFS server 200 are a kernel 202, a mount daemon 204, a dfstab file 206, and a share table file 208. As will be appreciated by those familiar with the NFS computing environment, the dfstab file 206 is a text file listing both the resources that the NFS server 200 is making available for sharing, the clients allowed to access the shared resources, and the access status of such clients. The share table file 208 is generated from the dfstab file 206 and provides similar information, but in a format more useful to the mount daemon 204.

As will be appreciated, the kernel 200 implements the more primitive functions of the server's operating system which in the NFS paradigm 200 includes an NFS service 220. As described below with reference to Figs. 9-13, the NFS service 220 manages all NFS requests. In order to enable such management, the NFS service 220 includes an export information table 222 and may include export authentication information such as an export authentication cache 224 for a specific client stored in cache memory. The export information table 222 provides information regarding the global access status to listed resources. That is, any access status provided in the export information table 222 applies to all NFS clients.

According to one embodiment of the present invention, the export information table 222 has entries such as entry 226 having a file system identifier 228, a read only (ro) bit 230, and a read-write (rw) bit 232. The file system identifier 228 may take any suitable form such as a file path. The ro bit 230 is set when all clients have read only access status with regards to the resource represented by the file system identifier 228. Similarly, the rw bit 232 is set when all clients have read and write access status with regards to the resource represented by the file system identifier 228. The ro bit 230 and the rw bit 232 are exclusive; only one of the two may be set. Of course, the ro bit 230 and the rw bit 232 may be implemented by another format representing equivalent information. For example, the ro bit 230 may be an ASCII string wherein the value "TRUE" indicates that all clients have read only access status.

The export authentication cache 224 provides information regarding an access status of a specific client. In the embodiment of Fig. 5, the export authentication cache 224 has a client identifier 240, a file system identifier 242, and a client access status 244. By way of example, the client identifier 240 may be a network source address, the file system identifier 242 may be a file path or other suitable identifier, and the client access status 244 may be a parameter indicating one of no access, read only access, or read and write access. As will be appreciated, the client access status 244 indicates the access status of the NFS client 12 with respect to the resources identified by the file system identifier 242.

In the embodiment of Fig. 5, the mount daemon 204 includes a mount service 250 and an NFS authentication service 252. As will be appreciated, a daemon is an autonomous process. In essence, a process within a computer has at least one thread of execution as well as exclusively allocated memory. The mount service 250 autonomously answers file system mount requests, making available those file systems which the NFS server 200 is willing to share. When the NFS client 12 attempts to mount a given file system 30, the mount service 250 authenticates that the NFS client 12 is entitled to access the given file system 30 and, if so, returns a file handle 24 corresponding to the particular file system. As will be appreciated, the mount service 250 essentially implements the functionality of the mount daemon 18 of the prior art.

The NFS server 200 also includes a dynamic NFS client authentication service 270. In embodiments such as that of Fig. 5, the dynamic NFS client authentication service 270 includes the NFS service 220 and the NFS authentication service 252. According to the present invention, for each NFS request 22, the dynamic NFS client authentication service authenticates the requesting NFS client 12. The steps involved in authenticating the NFS client 12 may include the following.

Initially an NFS request 22 including a file handle 24 and a file operation 26 is received. Then, a client's access status for a given file system 30 indicated by the file handle 24 is determined. The criteria for determining the client's access status may vary, but a fundamental criterion is the client's access status for the given file system 30 as provided in the share table file 208. However, this information may also be provided (directly or indirectly) in the export information table 222 or as an entry in the cache 224; in which case, the share table file 208 need not be consulted. Beyond this fundamental criterion, the client's access status may be further limited or expanded by other parameters.

For example, in some embodiments it may be desirable to limit access to certain resources during peak usage periods. A commercial on-line service may impose a hierarchy in its membership structure. The lowest level members would only have access to high demand resources during non-peak usage periods. In contrast, the highest level members access would never be limited. Another criterion which would be suitable for controlling access would be a current status of the given file system 30. For example, if the given file system 30 was currently off line, it may be desirable to limit access even though the NFS server 200 originally intended to share the file system 30. Accordingly, such information would be utilized by the dynamic NFS client authentication service 270 when authenticating the NFS client 12.

In any event, once the client's access status for the given file system is determined as one of no access, read only access, or read-write access, the authentication process continues by determining the nature of the file operation 26. For example, the file operating may be a read or write operation. Then, the client's access status is compared with the nature of the file operation 26 in order to determine if the file operation 26 should be executed. For example, if the file operation 26 requires modifying the given file system 30 but the client's access status is read only access, then the file operation 26 is unauthorized and will not be executed.

As seen in Fig. 5, preferred embodiments of the dynamic NFS client authenticating service 270 are implemented by multiple components. One rationale for providing only a portion (the NFS service 220) of the dynamic NFS client authenticating service 270 within the kernel is as follows. The kernel 200 is typically implemented in precious (in terms of cost and availability) random access memory such as RAM 134. As will be appreciated, the most time efficient response would arise from implementing the entire dynamic NFS client authenticating service 270 within the kernel. However, the costs of utilizing RAM 134 for the NFS service 220 must be balanced with the need for conserving RAM 134 for other software running on the NFS server 200.

In essence, the NFS service 220 ought to provide a minimal dynamic NFS client authentication. This includes the capability to (a) authenticate an NFS client 12 when the client's access status for a given file system 30 has been determined in a previous NFS request 22, (b) authenticate the NFS client 12 when the NFS server 200 provides read only access to all NFS clients for the given file system 30 and the file operation 26 does not require modifying the given file system 30, (c) authenticate the NFS client 12 when the NFS server 200 provides read-write access to all NFS clients for the given file system 30, and (d) make a dynamic authentication request to a resource external to the kernel 200 when none of the necessary conditions in (a)-(c) are met. Thus, the NFS authentication service 252 must be able to receive, perform, and reply to dynamic NFS client authentication requests sent from the NFS service 220. One suitable embodiment for separating the functionality of the dynamic NFS client authentication service 270 is described below in more detail with reference to Figs. 10-13.

Also shown in Fig. 5 is a network name service 260. As will be appreciated by those skilled in the art, a network name service 260 provides information about computers connected to the network 100. Of particular relevance to the present invention, the network name service 270 is operable to convert the NFS client 12's network source address into a hostname. This may be necessary since the typical share table file 208 is organized by hostnames, while the typical NFS request 22 only indicates the NFS client 12's network source address. This will be discussed below in more detail with reference to Fig. 12.

Turning next to Fig. 6, a initialization method 298 for an NFS server 200 in accordance with one embodiment of the present invention will now be described. The initialization method 298 begins in a step 300 by starting the NFS server 200. A number of steps not directly related to the present invention must be performed in order to bring the NFS server 200 into an operating state. However, these are well understood by those of skill in the art and, hence, no description is provided herein. A next step 302 processes the dfstab file 206 creating the share table file 208 and, internal to the kernel 202, the export info table 222.

After the dfstab file 206 has been processed, a step 304 starts the mount service 250 within the mount daemon 204. Then, a step 306 starts the NFS authentication service 252 within the mount daemon 204. These steps 304 and 306 may be performed in reverse order. Further, as will be appreciated, other embodiments of the present invention may suitably implement the mount service 250 and the NFS authentication service 252 within separate processes or within the kernel 202. Then, a step 308 starts the NFS service 220 within the kernel 202. Once the NFS service 220 is started, in a step 310 the NFS server 200 is ready to process NFS mount and file access requests.

With reference to Fig. 7, an authentic client response method 400 in accordance with one embodiment of the present invention will now be described. The method 400 begins in a step 402 where any required initialization procedures are performed. If not yet performed, the initialization procedures include those described above with reference to Fig. 6. Next, in a step 404, an NFS client 12 makes an NFS request 22 having a file handle 24 and a file operation 26. As will be appreciated, valid NFS requests include file operations such as read, delete, and modify. In response to the NFS request 22 (and in contrast to the prior art), the NFS server 200 will dynamically authenticate the NFS client 12. That is, the NFS server 200 will determine whether the NFS client 12 has the required access status to perform the file operation 26 upon a given file system 30 identified by the file handle 24. One suitable method for the NFS server 200 to dynamically authenticate the NFS client 12 is described below with reference to Fig. 9. In Fig. 7 the NFS client 12 is authenticated and thus the NFS server 200 implements the file operation 26. Accordingly, in a step 406, the NFS client 12 receives back a desired response 28.

Now, turning to Fig. 8, an unauthenticated client response method 410 in accordance with another aspect of the present invention will be described. The method 410 is initiated in step 412 and in a step 414 an NFS client 12 makes an NFS request 22 including a valid file handle 24 and a file operation 26. However (in contrast to the prior art), the NFS server 200 dynamically determines that the NFS client 12 does not have the access status required to perform the requested file operation 26. One suitable method for the NFS server 200 to dynamically authenticate the NFS client 12 is described below with reference to Fig. 9. Accordingly, the NFS server 200 does not perform the requested file operation 26. Instead of receiving the desired response, in a step 416 the NFS client 12 receives an error indication. Hence the NFS server 200's security is not breached.

With reference to Fig. 9, a method 430 for performing dynamic NFS client authentication in accordance with one aspect of the present invention is now described. The method 430 begins in a step 432 which includes any required initialization processes. As will be appreciated, these include network initialization as well as starting a mount daemon 204 and an NFS service 220. Of course, in general these initialization processes need only be done once and subsequent instances of the method 430 would not include such steps.

In a next step 434, the NFS server 200 receives an

NFS request 22 from an NFS client 12. As described above with reference to Fig. 5, the NFS request 22 includes a file handle 24 and a file operation 26. As will be appreciated, the file handle 24, if valid, represents a given file system 30 present on the NFS server 200. The file operation 26 is an operation which may be performed on the given file system 30. In response to the request 22, in a step 434 the NFS server 200 compares the client's access status with the access status required to perform the file operation 26 and responds accordingly. For example, if the client had read only access status and the file operation 26 required modifying the given file system 30, the NFS server 200 could respond with an error message informing the NFS client 12 that the required write access status was lacking. The NFS server 200 could also respond by indicating that the requested command could not be performed at this time.

In some implementations, the NFS server 200 may respond to inauthentic NFS clients with more severe security measures. By way of example, the NFS server 200 may record in a file and/or on a system terminal that an unauthenticated NFS request 22 was received from NFS client 12. Depending upon the circumstances, the NFS server 200 may determine that the NFS client 12 is attacking and preclude the NFS server 12 from making further NFS requests. One embodiment of step 436 will be described below in more detail with reference to Fig. 10.

Turning next to Fig. 10, a method for performing step 436 of Fig. 9 in accordance with one embodiment of the present invention will now be described. The method begins in step 452 where the NFS service 220 receives and begins responding to the NFS request 22 which the NFS server 200 received in step 434 of Fig. 9. As described above in reference to Fig. 5, the NFS service 220 is implemented within the kernel 202 of the NFS server 200. As will be appreciated, if the format of the NFS request 22 is not suitable for use by the NFS service 220, step 452 may include processing the NFS request 22 to make it suitable for use by the NFS service 220. In general, this processing is done external to the kernel 202. By way of example, data is often marshaled into a format suitable for network transmission, then transmitted over the network. Thus upon receipt of the NFS request, it may be necessary to unmarshal the NFS request 22 prior to utilization by the NFS service 220. However, the network format may be suitable for utilization by the NFS service 220. Of course, these are application specific details which will be familiar to those skilled in the art.

Once the NFS service 220 has a suitably formatted NFS request 22, a search step 454 searches in the export information table 222 for the given file system 30. According to the embodiment described above with reference to Fig. 5, the export information table 222 has entries such as entry 226 having a file system identifier 228, a read only (ro) bit 230, and a read-write (rw) bit 232. In preferred embodiments, the file system identifier 228 is in a format identical to the format of the file handle 24. Thus search step 454 utilizes the file handle 24 as a key to locate the given file system 30 in export information table 222.

Once the export information table 222 has been searched in step 454, a step 456 determines whether the given file system 30 was found in the export information table 222. The given file system 30 is only present in the export information table 222 when the NFS server 200 is making the given file system 30 accessible for sharing. When the given file system 30 is not found in search step 454, control is passed to a step 458 which returns an error message to the NFS client 12. In some embodiments of the present invention, additional or different security measures may be performed. As described above with reference to Fig. 9, these include logging a message on the system terminal, maintaining a file record of unauthenticated client requests, and/or precluding operation of future NFS requests by the NFS client 12.

When search step 454 successfully finds the given file system 30, control is passed from determination step 456 to a step 460. Step 460 calls a subroutine NFS AUTH in order to determine the client's access status. The parameters for the call of step 460 include the client's network source address and the information from the export information table entry 226 corresponding to the given file system 30 (found in step 454). The client's network source address is a numerical identifier of the NFS client 12's network address. For example, if the network is operating under the well known TCP/IP network protocol, then the client's network source address will be the client's Internet protocol (IP) address. One suitable embodiment of subroutine NFS AUTH will be described in detail below with reference to Fig. 11. As will be familiar to those skilled in the art, a subroutine is a portion of computer code which performs a process required at multiple points of execution within the computer code. By implementing such a process via a subroutine, redundancy in the computer code is minimized. However, other suitable embodiments of the present invention may well implement redundant code rather than making calls to a subroutine NFS AUTH.

In any event, in response to the call of step 460, the NFS service 220 receives the client's access status from the subroutine NFS AUTH. As will be appreciated, the client's access status will be one of read only (ro) access, read and write (rw) access, or no access. Control is then passed to a step 464 which determines whether the client's access status is equal to no access. If so, control is passed from step 464 to a step 466 which returns an error message to the NFS client 12. As will be appreciated, other embodiments may perform additional security measures. When the client's access status is something other than no access, control is instead passed to a step 468 which determines whether the requested file operation 26 requires a modification to the given file system 30. When the requested file operation

26 does not require a modification, then in a step 470 the NFS client 12 is provided ro access status and the requested file operation 26 is performed. As will be appreciated, in the case when the requested file operation 26 does not require a modification and the NFS client 12 has an access status other than no access, it is sufficient in step 470 to provide merely ro access status to perform the requested file operation 26.

However, when step 468 determines that the requested file operation 26 requires a modification to the given file system 30, control is passed to a step 472 where the subroutine NFS AUTH is called (again) with the client's network source address and the information from the export information table entry 226 as parameters. This is necessary because, according to the embodiment of Fig. 10, the client's access status is not saved from step 464 to step 472. This is done in order to minimize use of the kernel 202's memory space. However, in other embodiments, kernel 202 could maintain a copy of the client's access status in which case step 472 could be skipped. In any event, a next step 474 determines whether the client's access status is equal to ro access. When the client's access status is ro access, then the NFS client 12 is not authorized to perform the requested file operation 26 because it has been determined, in step 468, to require a modification to the given file system 30. Accordingly, control is passed to a step 476 which returns an error message to the NFS client 12. Of course, step 476 may implement other appropriate security measures including those described above with reference to step 458. When step 474 determines that the client's access status is not equal to ro, then the client's access status is rw. Thus, a next step 478 provides the NFS client 12 with ro access and performs the requested file operation 26 on the given file system 30.

As will be appreciated, the control flow of the method 10 could be rearranged in a variety of ways, each suitable to determine whether the client's access status for the given file system 30 satisfied the access required for the requested file operation 26.

Turning now to Fig. 11, a method 500 for the subroutine NFS AUTH to respond to a call requesting the access status of an NFS client 12 in accordance with one aspect of the present invention will now be described. As will be appreciated, the method 500 may be incorporated into an NFS service 220 which does not utilize subroutines in order to determine the access status of the NFS client 12. In any event, the method 500 begins in a step 502 which performs any required initialization processes. Then in a step 504 a request for access status having the NFS client 12's information and the file system identifier 228 as parameters is received. A next step 506 determines if the ro bit 230 is set. If so, control is passed to a step 508 which returns an access status of ro for the NFS client 12. Otherwise, a step 514 determines if the rw bit 232 is set and, if so, control is passed to a step 516 which returns an access status of rw for the NFS client 12.

When neither the ro bit 230 nor the rw bit 232 is set for the given file system 30, then a step 520 determines whether the client's source network address is found within an export authentication cache 224. As described above with respect to the embodiment of Fig. 5, each export authentication cache such as export authentication cache 224 is stored in the kernel 202's cache memory and provides information regarding an access status of a specific NFS client. In the embodiment of Fig. 11, each export authentication cache 224 includes a source network address 240, a file system identifier 242, and a client access status 244. As will be understood, the client access status 244 indicates the access status of the NFS client 12 with respect to the resources identified by the file system identifier 242.

When step 520 matches the client's source network address with the source network address 240 in a particular export authentication cache 224, a step 522 returns the value in the corresponding client access status 224 directly and the method 500 is done at step 530. Step 530 then passes control back to the main code of the NFS server 220. In the case where the export authentication cache 224 corresponding to both the given file system 30 and the NFS client 12 is not yet created, control is passed to a step 524. In step 524, the NFS AUTH subroutine calls the NFS authentication service 252, passing the client's network source address and the file handle 24 as parameters. As described above with reference to Fig. 5, in preferred embodiments the NFS authentication service 252 is resident in the mount daemon 204. However, in other suitable embodiments the NFS authentication service 252 could be residing in a separate process, even the kernel 202.

In response to the call of step 524, the NFS AUTH subroutine receives the client's access status with respect to the given file system 30 in a step 526. Then, in a step 528, the NFS AUTH subroutine creates a corresponding entry of export authentication cache 224. Because of this, subsequent queries as to the NFS client 12's access status can be answered directly from the cache memory 224 in the kernel 202. Once the authentication cache 224 is created, control passes to step 522 which returns the NFS client 12's access status directly from cache. Then, in step 530, subroutine NFS AUTH is complete and process control is passed back to the main code of the NFS server 220.

Turning now to Fig. 12, a method 550 for an NFS authentication service 252 to determine an NFS client 12's access status to a given file system 30 in accordance with yet another aspect of the present invention will be described. As described above with reference to Fig. 5, the NFS authentication service 252 resides outside the kernel 202 and within the mount daemon 204. This is done merely as a sound computer programming practice. However, in other embodiments of the present invention, the NFS authentication service 252 could reside in a separate process or even within the kernel. This is primarily an application specific detail which may be decided upon implementation of the present invention. The method 550 starts in a step 552 where any necessary initialization processes are performed. Then in a step 554, the NFS authentication service 252 receives a request from the kernel to determine the NFS client 12's access status. A next step 556 searches the share table file 208 to determine if the given file system 30 has an entry therein. If the given file system 30 does not have an entry, then control passes to a step 558 which returns an access status of no access to the kernel 202. In some embodiments, record may be logged of this inconsistency in a file and/or on the system terminal. Once step 558 is done, control is passed to a step 568 where the current instance of the method 550 is complete.

When the given file system 30 does have an entry in the share table file 208, a step 560 calls a network name service 270 to determine the hostname corresponding to the network source address of the NFS client 12. As described above with reference to Fig. 5, the network name service 270 performs network services such as providing a hostname corresponding to a given network source address. Because, in general, the share table file 208 identifies NFS clients by hostnames rather than network source addresses, step 560 is required to enable searching the share table file 208. However, in embodiments where the share table file 208 identifies NFS clients by their network source addresses, step 560 would be unnecessary. In response to step 560, a step 562 receives the hostname associated with the NFS client 12. Then a step 564 searches the share table file 208 to determine the access status of the NFS client 12 for the given file system 30 using the hostname associated With the NFS client 12. As will be appreciated, when an access status is not found for the given file system 30, it merely indicates that the NFS client 12 has a status of no access. Once the access status for the NFS client 12 is determined, a step 566 returns the access status to the kernel 202.

In the embodiment of Fig. 12, the NFS authentication service 252 determined the access status of the NFS client 12 according to the share table file 208. However, in accordance with other embodiments of the present invention, an NFS authentication service 252 can utilize additional resources and/or strategies in determining whether the NFS client 12 is entitled to access a particular file system 30. For example, an NFS server 200 may limit access to certain resources during peak use periods, allowing only a select group or a finite number of clients access during such times. This could be implemented by providing the NFS authentication server 252 the current time and a table of clients authorized for certain resources during the peak periods.

With reference to Fig. 13, a method 600 for temporarily modifying the access status of an NFS client 12 in accordance with one aspect of the present invention will be described. In addition to providing dynamic NFS client authentication, the teaching of the present invention enables modification of the access status of an NFS client 12 after the NFS client 12 mounts a given file system 30. A step 602 begins the temporary modification which is typically initiated and performed by a system administrator of an NFS server 200. In a step 604, a share table file 208 is modified in accordance with the desired changes in access status for NFS clients. Then a step 606 replaces corresponding entries in an export information table 222 to represent the modified access status. Step 606 also includes purging of the cache 224 entries. The method 600 is then complete in step 608. As will be appreciated, a proper combination of NFS share commands 610 will implement the steps 604 and 606.

The method 600 of Fig. 13 can be adapted to permanently modify the access status of the NFS client 12 by performing modifications equivalent to those made to the share table file 208 on the dfstab file 206. If such modifications are performed, then upon initialization of the NFS server 200, these changes will automatically become part of the share table file 208 and the export information table 222, as described above with reference to Fig. 6.

Although only one embodiment of the present invention has been described, it should be understood that the present invention may be embodied in many other specific forms without departing from the spirit or scope of the invention. For example, the concepts described herein are equally applicable within a variety of distributed file system computing environments. Therefore, the described embodiments should be taken as illustrative and not restrictive, and the invention should be defined by the following claims and their full scope of equivalents.

Claims

1. A method implemented on a server computer system for providing dynamic client authentication in a distributed file system computing environment, the method comprising the computer controlled steps of:

    receiving an NFS request from an NFS client, the NFS request including a file handle representing a given file system available on the server computer system and a file operation to be performed upon the given file system, the given file system modifiable by clients of the server computer having a corresponding access status of read-write with respect to the given file system, readable by clients of the server computer having the corresponding access status of read only with respect to the given file system, and inaccessible to all other clients of the server computer;

    dynamically determining whether the NFS client has an access status sufficient to perform the NFS request; and

performing the NFS request when the NFS client has sufficient access status.

2. A method as recited in claim 1 wherein the step of determining whether the NFS client has the access status sufficient to perform the NFS request includes the substeps of:

searching an export information table resident on the server computer system to determine whether the given file system has an entry therein; and

returning an error indication to the NFS client when the file system is not found in the export information table.

3. A method as recited in claim 2 wherein when the export information table has an entry for the given file system, the entry including a read only bit which when set indicates global read only access to the given file system and a read-write bit which when set indicates global read and write access to the given file system, the read only bit and the read-write bit being exclusive, the step of determining whether the NFS client has an access status sufficient to perform the NFS request further including the substeps of:

when the read only bit is set, setting the client's access status to read only; and

when the read-write bit is set, setting the client's access status to read-write.

4. A method as recited in claim 3 wherein when the client's access status is one of read only and read-write and the file operation does not require a modification of the given file system, the client's access status is sufficient to perform the NFS request.

5. A method as recited in claim 3 wherein when the client's access status is read only and the file operation requires a modification of the given file system, the client's access status is not sufficient to perform the NFS request.

6. A method as recited in claim 3 wherein when the client's access status is read-write, the client's access status is sufficient to perform the NFS request.

7. A method as recited in claim 3 wherein when neither the read only bit nor the read-write bit is set, the method further includes the steps of:

searching a cache memory resident on the server computer system to find a specific export authentication cache entry for the NFS client which corresponds to the given file system, the specific export authentication cache entry, when it exists, indicating the client's access status to the given file system; and

when the specific export authentication cache entry does not exist, creating the specific export authentication.

8. A method as recited in claim 7 further including the steps of:

setting the client's access status to that indicated by the specific export authentication cache;

when the client's access status is one of read only and read-write and the file operation does not require a modification of the given file system, determining that the client's access status is sufficient to perform the NFS request;

when the client's access status is read only and the file operation requires a modification of the given file system, determining that the client's access status is not sufficient to perform the NFS request; and

when the client's access status is read-write, determining that the client's access status is sufficient to perform the NFS request.

9. A method as recited in claim 7 wherein the step of creating the specific export authentication includes the substeps of:

searching a share table file resident on the server computer system to find a share entry for the given file system;

setting the client's access status to no access when the share entry for the given file system is not found in the share table file;

determining the client's access status from the share entry for the given file system when the share entry is found in the share table file; and

setting the client's access status according to the share entry for the given file system when the share entry is found in the share table file.

10. A method as recited in claim 9 wherein access status information is stored in the share table file according to client network names and the substep of determining the client's access status from the

share entry for the given file system includes calling a network name service available within the NFS computing environment in order to ascertain a network name for the NFS client.

11. A method as recited in claim 9 wherein the client's access status can be dynamically modified, without necessitating the NFS client to dismount, by modifying both the share table file and the export information table to indicate the client's modified access status.

12. A method as recited in claim 1 wherein the given file system is a resource available on the server computer, the resource being selected from the group including a file and a file system hierarchical structure.

13. A method as recited in claim 1 wherein the step of determining whether the NFS client has the access status sufficient to perform the NFS request includes consideration of at least one of time, date, identity of the NFS client, a nature of the NFS request, and a current status of a resource which the NFS request operates upon.

14. A computer readable medium containing a computer program for providing dynamic client authentication to a server computer operating in a distributed file system computing environment, the computer program comprising computer executable instructions for:

receiving an NFS request from an NFS client, the NFS request including a file handle representing a given file system available on the server computer system and a file operation to be performed upon the given file system, the given file system modifiable by clients of the server computer having a corresponding access status of read-write with respect to the given file system, readable by clients of the server computer having the corresponding access status of read only with respect to the given file system, and inaccessible to all other clients of the server computer;

dynamically determining whether the NFS client has an access status sufficient to perform the NFS request; and

performing the NFS request when the NFS client has sufficient access status.

15. A computer readable medium as recited in claim 14 wherein the computer executable instruction of determining whether the NFS client has the access status sufficient to perform the NFS request includes subinstructions for:

searching an export information table resident on the server computer system to determine whether the given file system has an entry therein; and

returning an error message to the NFS client when the file system is not found in the export information table.

16. A computer readable medium as recited in claim 15 wherein the export information table has an entry for the given file system, the entry including a read only bit which when set indicates global read only access to the given file system and a read-write bit which when set indicates global read and write access to the given file system, the read only bit and the read-write bit being exclusive, and the computer program instruction for determining whether the NFS client has an access status sufficient to perform the NFS request further includes the computer executable subinstructions of:

setting the client's access status to read only when the read only bit is set; and

setting the client's access status to read-write when the read-write bit is set.

17. A computer readable medium as recited in claim 16 further including computer executable instructions such that when the client's access status is one of read only and read-write and the file operation does not require a modification of the given file system, the client's access status is sufficient to perform the NFS request.

18. A computer readable medium as recited in claim 16 further including computer executable instructions such that when the client's access status is read only and the file operation requires a modification of the given file system, the client's access status is not sufficient to perform the NFS request.

19. A computer readable medium as recited in claim 16 further including computer executable instructions such that when the client's access status is read-write, the client's access status is sufficient to perform the NFS request.

20. A computer readable medium as recited in claim 16 further including computer executable instructions such that when neither the read only bit nor the read-write bit is set, the computer program further executes the computer instructions for:

searching a cache memory resident on the server computer system to find a specific export authentication cache entry for the NFS client which corresponds to the given file system, the specific export authentication cache entry, when it exists, indicating the client's access status to the given file system; and

when the specific export authentication cache entry does not exist, creating the specific export authentication.

21. A computer readable medium as recited in claim 7 further including computer program instructions for:

setting the client's access status to that indicated by the specific export authentication cache;

when the client's access status is one of read only and read-write and the file operation does not require a modification of the given file system, determining that the client's access status is sufficient to perform the NFS request;

when the client's access status is read only and the file operation requires a modification of the given file system, determining that the client's access status is not sufficient to perform the NFS request; and

when the client's access status is read-write, determining that the client's access status is sufficient to perform the NFS request.

22. A computer readable medium as recited in claim 20 wherein the computer program instruction for creating the specific export authentication includes the computer executable subinstructions for:

searching a share table file resident on the server computer system to find a share entry for the given file system;

setting the client's access status to no access when the share entry for the given file system is not found in the share table file;

determining the client's access status from the share entry for the given file system when the share entry is found in the share table file; and

setting the client's access status according to the share entry for the given file system when the share entry is found in the share table file.

23. A computer readable medium as recited in claim 14 wherein the given file system is a resource available on the server computer, the resource being selected from the group including a file and a file system hierarchical structure.

24. A computer readable medium as recited in claim 14 wherein the computer program instruction for determining whether the NFS client has the access status sufficient to perform the NFS request considers at least one of time, date, identity of the NFS client, a nature of the NFS request, and a current status of a resource which the NFS request operates upon.

25. A server computer for use in a distributed file system computing environment, the server computer operable to provide dynamic NFS client authentication, the server computer comprising:

a central processing unit (CPU);

a random access memory accessible by the CPU;

a read only memory accessible by the CPU;

a network input/output port coupled with the CPU;

a mass storage device accessible by the CPU, the mass storage device capable of storing a given file system modifiable by clients of the server computer having an access status of read-write with respect to the given file system, readable by clients of the server computer having the access status of read only with respect to the given file system, and inaccessible to all other clients of the server computer;

a kernel implemented on the server computer, the kernel implementing primitive functions of an operating system for the server computer; and

a dynamic NFS client authentication service operable to receive an NFS request from an NFS client and to dynamically authenticate the NFS client in relation to the NFS request, the dynamic NFS client authentication service considering at least one of time, date, identity of the NFS client, a nature of the NFS request, and a current status of a resource which the NFS request operates upon.

26. A server computer as recited in claim 25 wherein the dynamic NFS client authentication service includes:

an NFS service implemented within the kernel, the NFS service operable to receive the NFS request from the NFS client, the NFS request including a file handle identifying the given file system and a file operation to be performed on the given file system, the client's access status for the given file system being one of no access, read only access, and read-write access, the NFS service also operable to (a) authenticate the NFS client when the client's access status for the given file system has been determined in a previous NFS request, (b) authenticate the NFS client when the server computer provides read only access to all NFS clients for the given file system and the file operation does not require modifying the given file system, (c) authenticate the NFS client when the server computer provides read-write access to all NFS clients for the given file system, and (d) make a dynamic authentication request to a resource external to the kernel when none of the necessary conditions in (a)-(c) are met; and

an NFS authentication service implemented on the server computer system and external to the kernel, the NFS authentication service being operable to receive, perform, and reply to dynamic NFS client authentication requests sent from the NFS service.

27. A server computer as recited in claim 26 wherein the kernel includes:

an export information table resident in the kernel, the export information table having entries for a plurality of file systems available on the server computer, each entry being identical in format, an entry for a specific file system including a read only bit which when set indicates global read only access to the specific file system and a read-write bit which when set indicates global read and write access to the specific file system, the read only bit and the read-write bit being exclusive; and

a cache memory for storing a plurality of export authentication cache entries, a particular export authentication cache entry including identifiers for a file system and an NFS client, and an access status of the NFS client with respect to a file system identified by the file system identifier.

28. A server computer as recited in claim 25 further including a share table file including a list of file systems available for sharing on the server computer and a corresponding plurality of client's access status.

29. A computer network including a plurality of computer systems, wherein a one of the plurality of computer systems is a server computer as recited in claim 25.

30. A method implemented on a server computer system for providing dynamic client authentication in a distributed file system computing environment, the method comprising the computer controlled steps of:

receiving an NFS request from an NFS client, the NFS request including a file handle representing a given file system available on the server computer system and a file operation to be performed upon the given file system, the given file system modifiable by clients of the server computer having a corresponding access status of read-write with respect to the given file system, readable by clients of the server computer having the corresponding access status of read only with respect to the given file system, and inaccessible to all other clients of the server computer;

searching an export information table resident on the server computer system to determine whether the given file system has an entry therein, the export information table having an entry for the given file system, the entry including a read only bit which when set indicates global read only access to the given file system and a read-write bit which when set indicates global read and write access to the given file system, the read only bit and the read-write bit being exclusive;

when the read only bit is set, setting the client's access status to read only;

when the read-write bit is set, setting the client's access status to read-write;

when neither the read only bit nor the read-write bit is set, performing the following substeps of:

(a) searching a cache memory resident on the server computer system to find a specific export authentication cache entry for the NFS client which corresponds to the given file system, the specific export authentication cache entry, when it exists, indicating the client's access status to the given file system to which the client's access status is then set; and

(b) when the specific export authentication cache entry does not exist, creating the specific export authentication cache entry and then setting the client's access status to that indicated by the newly created specific export authentication cache entry, the specific export authentication cache entry creation including:

(i) searching a share table file resident on the server computer system to find a share entry for the given file system;

(ii) setting the client's access status to no access when the share entry for the given file system is not found in the share table file;

(iii) determining the client's access status from the share entry for the given file system when the share entry is found in the share table file; and

(iv) setting the client's access status according to the share entry for the given file system when the share entry is found in the share table file; and

performing the NFS request when either (i) the client's access status is read only and the file operation does not require a modification of the given file system or (ii) the client's access status is read-write.

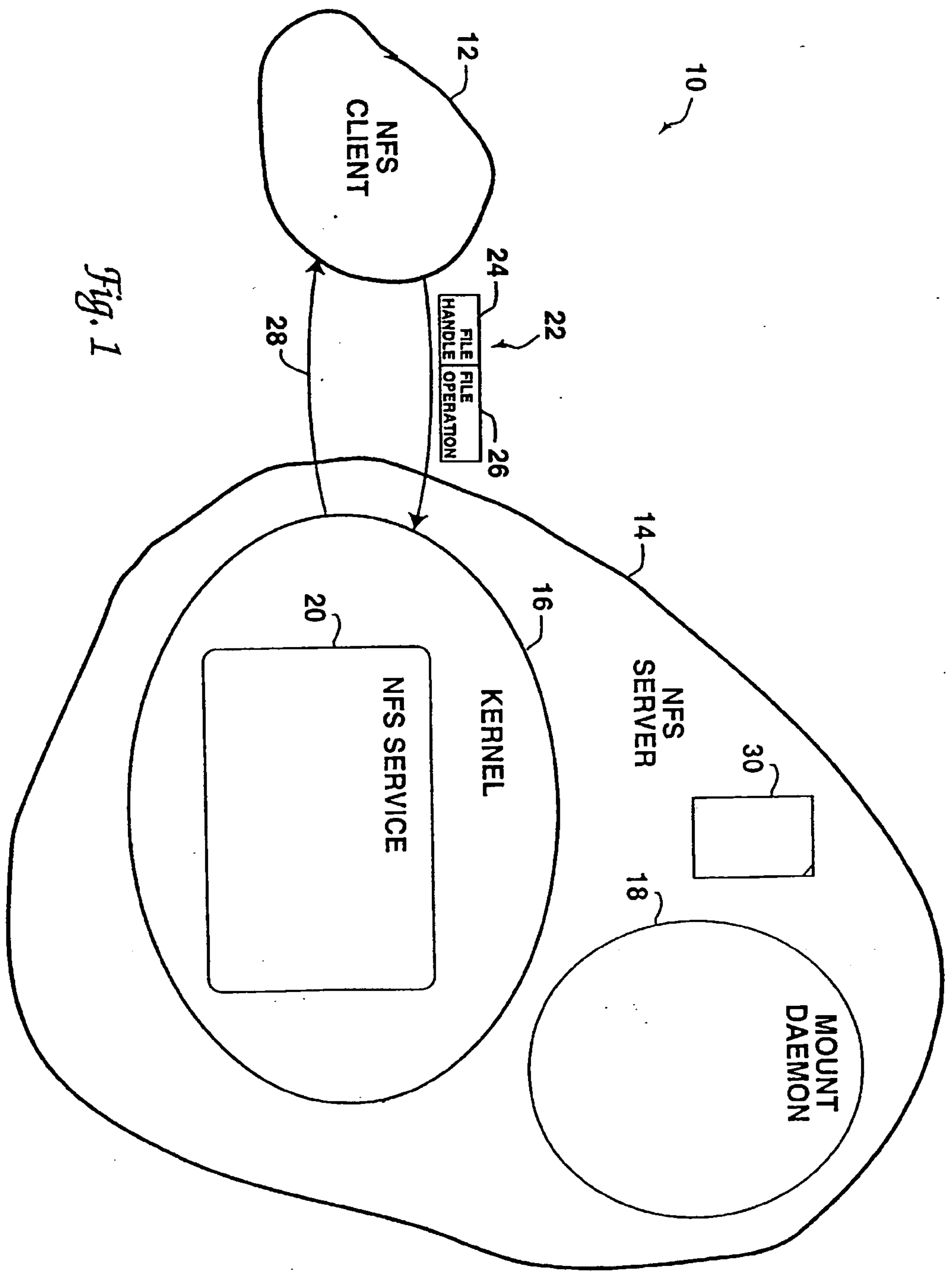

10
12
NFS CLIENT
22
24
FILE HANDLE
FILE OPERATION
26
28
14
NFS SERVER
16
KERNEL
20
NFS SERVICE
30
18
MOUNT DAEMON

Fig. 1

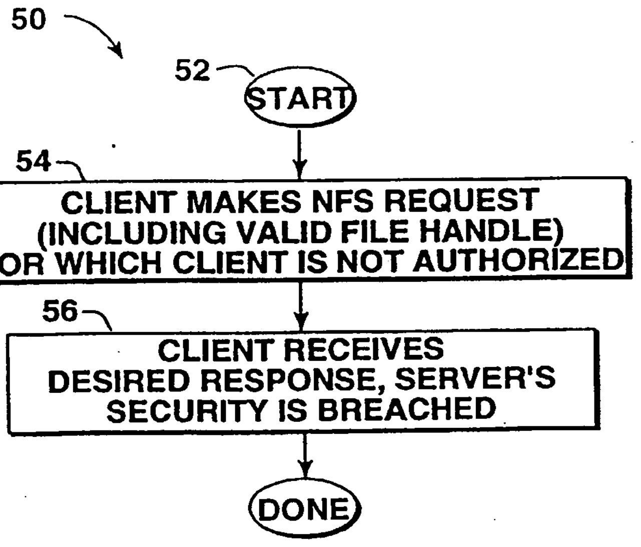

50
52
START
54
CLIENT MAKES NFS REQUEST (INCLUDING VALID FILE HANDLE) FOR WHICH CLIENT IS NOT AUTHORIZED
56
CLIENT RECEIVES DESIRED RESPONSE, SERVER'S SECURITY IS BREACHED
DONE

Fig. 2

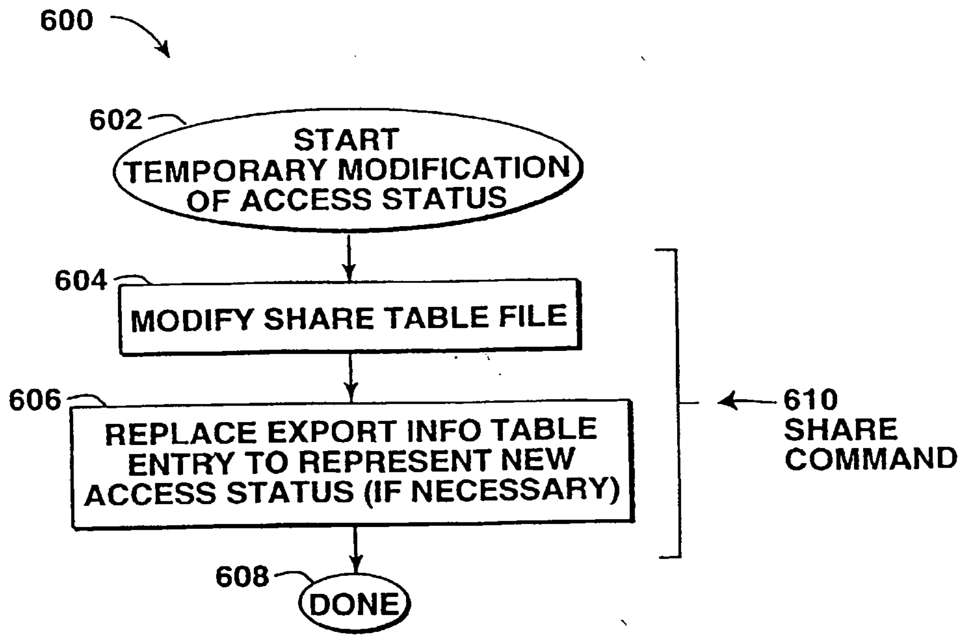

600
602
START TEMPORARY MODIFICATION OF ACCESS STATUS
604
MODIFY SHARE TABLE FILE
606
REPLACE EXPORT INFO TABLE ENTRY TO REPRESENT NEW ACCESS STATUS (IF NECESSARY)
610 SHARE COMMAND
608
DONE

Fig. 13

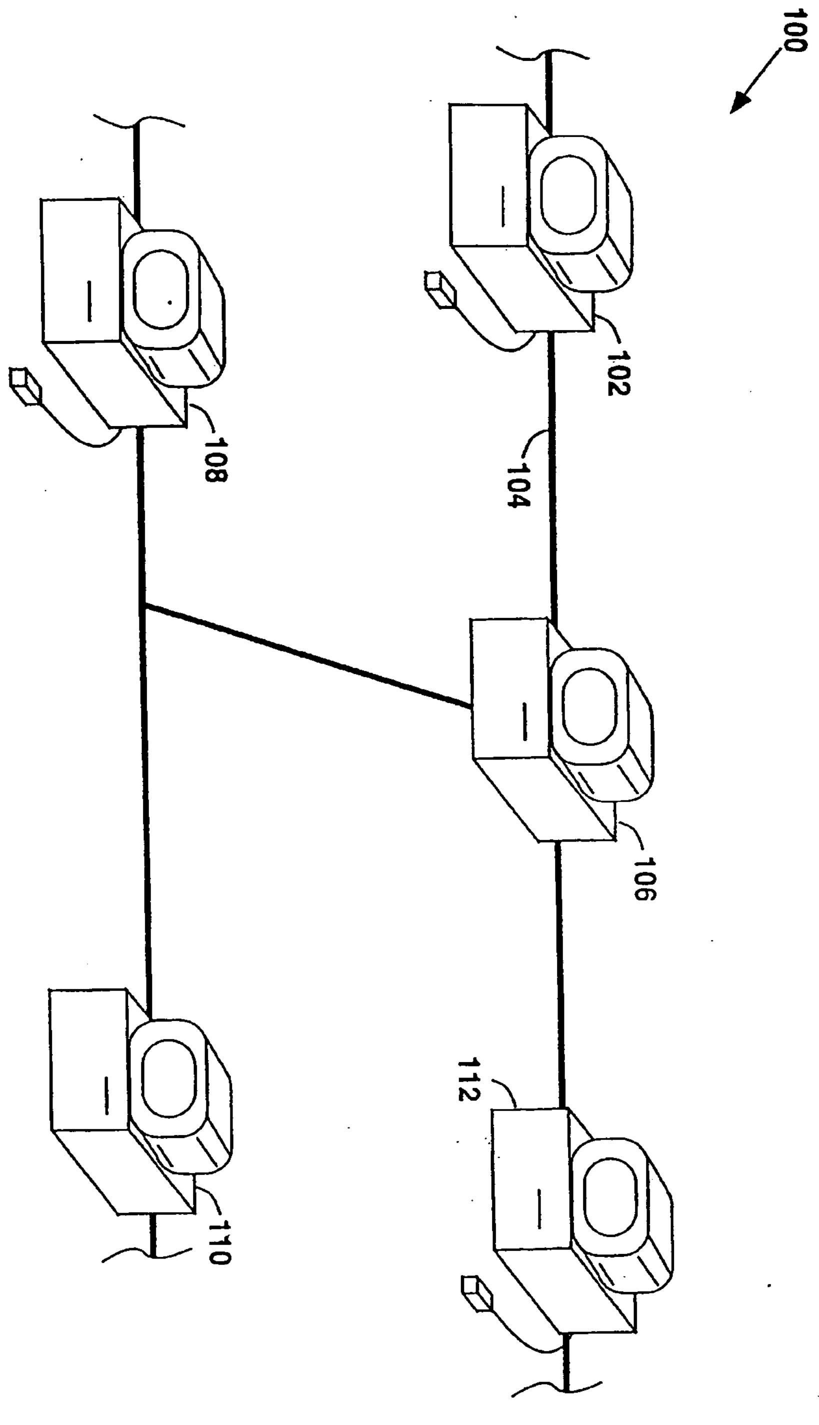


Fig. 3

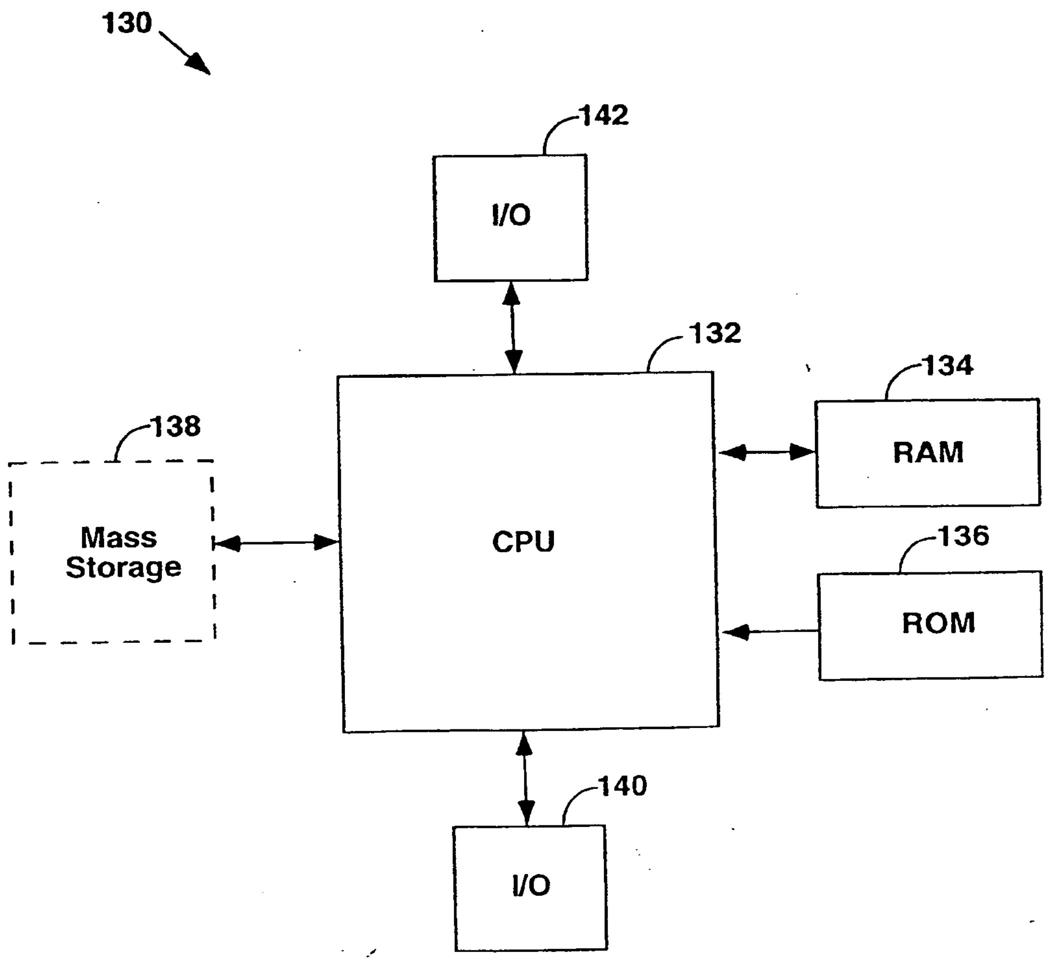


Fig. 4

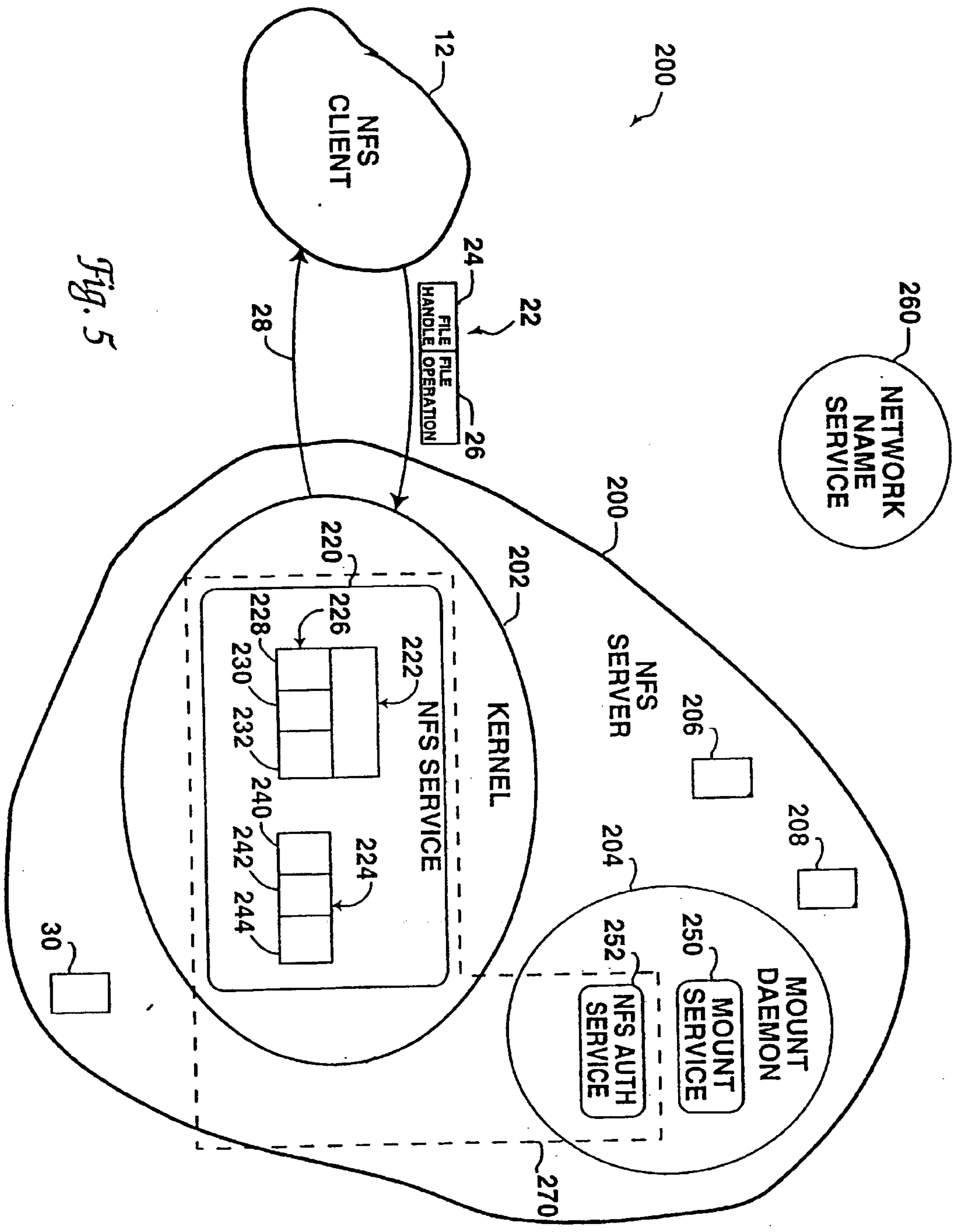

200
NFS CLIENT
12
22
24
FILE HANDLE
FILE OPERATION
26
28
260
NETWORK NAME SERVICE
200
202
NFS SERVER
KERNEL
NFS SERVICE
220
222
226
228
230
232
224
240
242
244
206
208
204
MOUNT DAEMON
250
MOUNT SERVICE
252
NFS AUTH SERVICE
30
270

Fig. 5

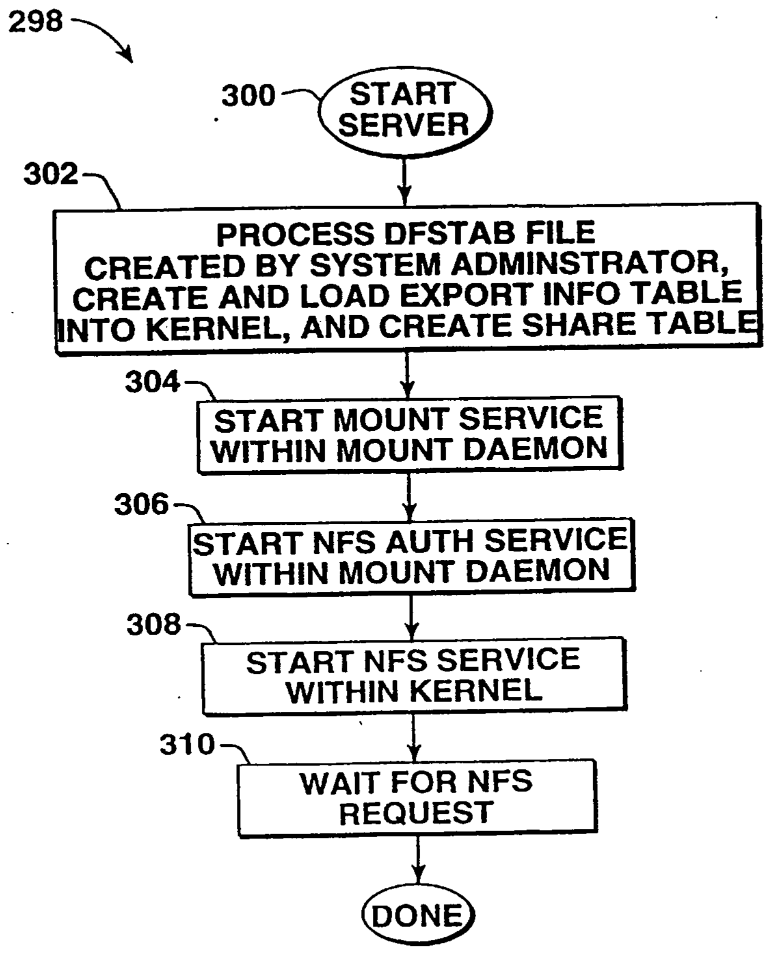

298
300
START SERVER
302
PROCESS DFSTAB FILE CREATED BY SYSTEM ADMINSTRATOR, CREATE AND LOAD EXPORT INFO TABLE INTO KERNEL, AND CREATE SHARE TABLE
304
START MOUNT SERVICE WITHIN MOUNT DAEMON
306
START NFS AUTH SERVICE WITHIN MOUNT DAEMON
308
START NFS SERVICE WITHIN KERNEL
310
WAIT FOR NFS REQUEST
DONE

Fig. 6

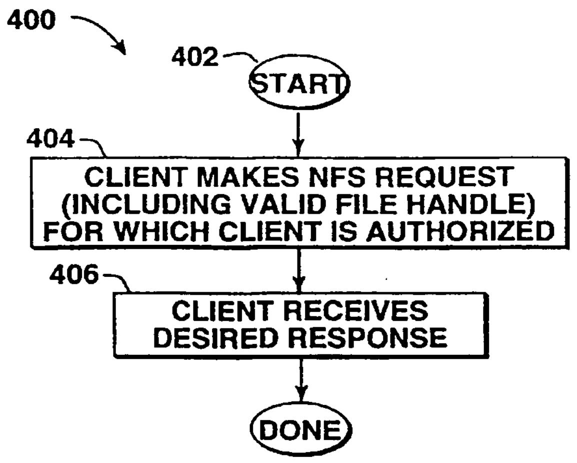

400
402
START
404
CLIENT MAKES NFS REQUEST (INCLUDING VALID FILE HANDLE) FOR WHICH CLIENT IS AUTHORIZED
406
CLIENT RECEIVES DESIRED RESPONSE
DONE

Fig. 7

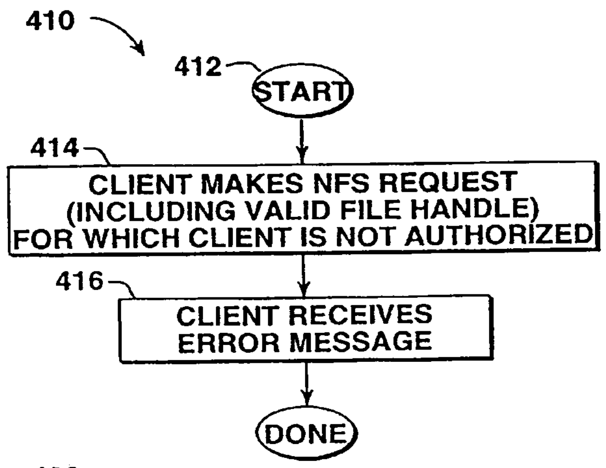

410
412
START
414
CLIENT MAKES NFS REQUEST (INCLUDING VALID FILE HANDLE) FOR WHICH CLIENT IS NOT AUTHORIZED
416
CLIENT RECEIVES ERROR MESSAGE
DONE

Fig. 8

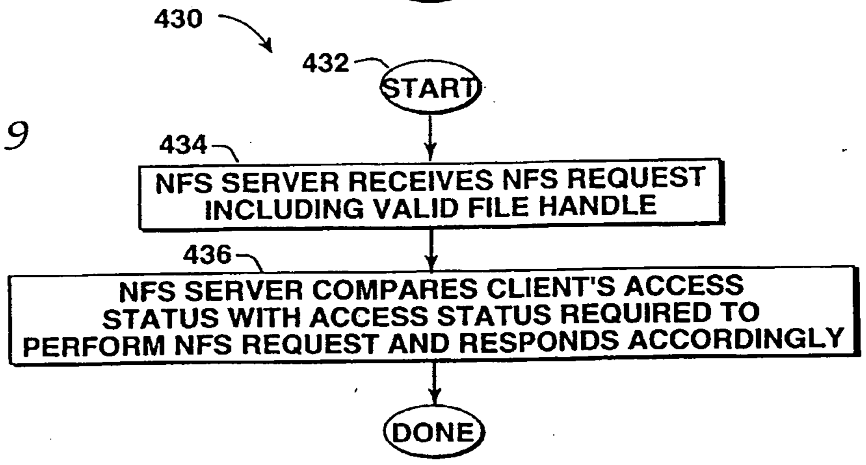

430
432
START
434
NFS SERVER RECEIVES NFS REQUEST INCLUDING VALID FILE HANDLE
436
NFS SERVER COMPARES CLIENT'S ACCESS STATUS WITH ACCESS STATUS REQUIRED TO PERFORM NFS REQUEST AND RESPONDS ACCORDINGLY
DONE

Fig. 9

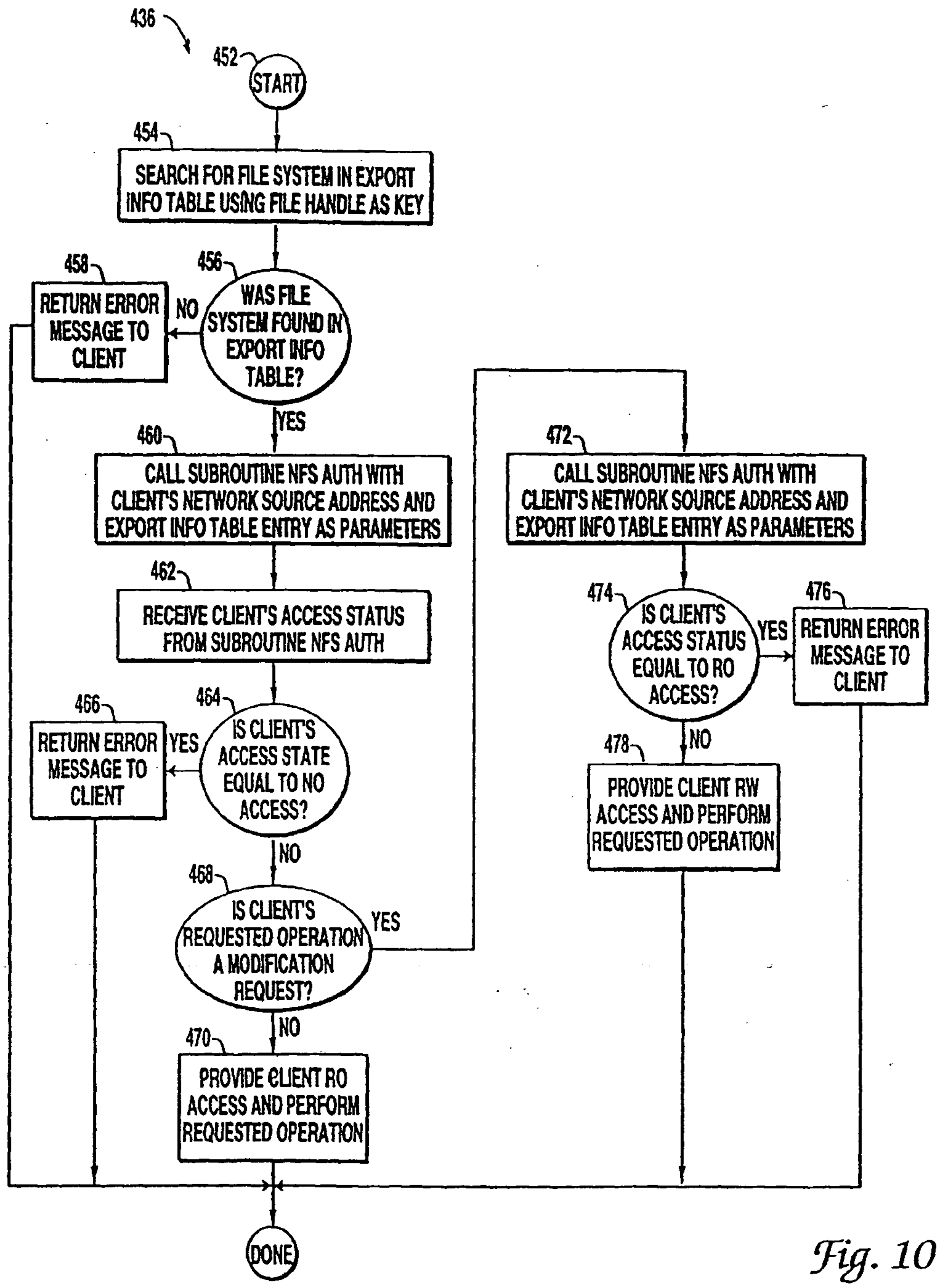

436
452
START
454
SEARCH FOR FILE SYSTEM IN EXPORT INFO TABLE USING FILE HANDLE AS KEY
456
WAS FILE SYSTEM FOUND IN EXPORT INFO TABLE?
NO
458
RETURN ERROR MESSAGE TO CLIENT
YES
460
CALL SUBROUTINE NFS AUTH WITH CLIENT'S NETWORK SOURCE ADDRESS AND EXPORT INFO TABLE ENTRY AS PARAMETERS
462
RECEIVE CLIENT'S ACCESS STATUS FROM SUBROUTINE NFS AUTH
464
IS CLIENT'S ACCESS STATE EQUAL TO NO ACCESS?
YES
466
RETURN ERROR MESSAGE TO CLIENT
NO
468
IS CLIENT'S REQUESTED OPERATION A MODIFICATION REQUEST?
YES
NO
470
PROVIDE CLIENT RO ACCESS AND PERFORM REQUESTED OPERATION
472
CALL SUBROUTINE NFS AUTH WITH CLIENT'S NETWORK SOURCE ADDRESS AND EXPORT INFO TABLE ENTRY AS PARAMETERS
474
IS CLIENT'S ACCESS STATUS EQUAL TO RO ACCESS?
YES
476
RETURN ERROR MESSAGE TO CLIENT
NO
478
PROVIDE CLIENT RW ACCESS AND PERFORM REQUESTED OPERATION
DONE

Fig. 10

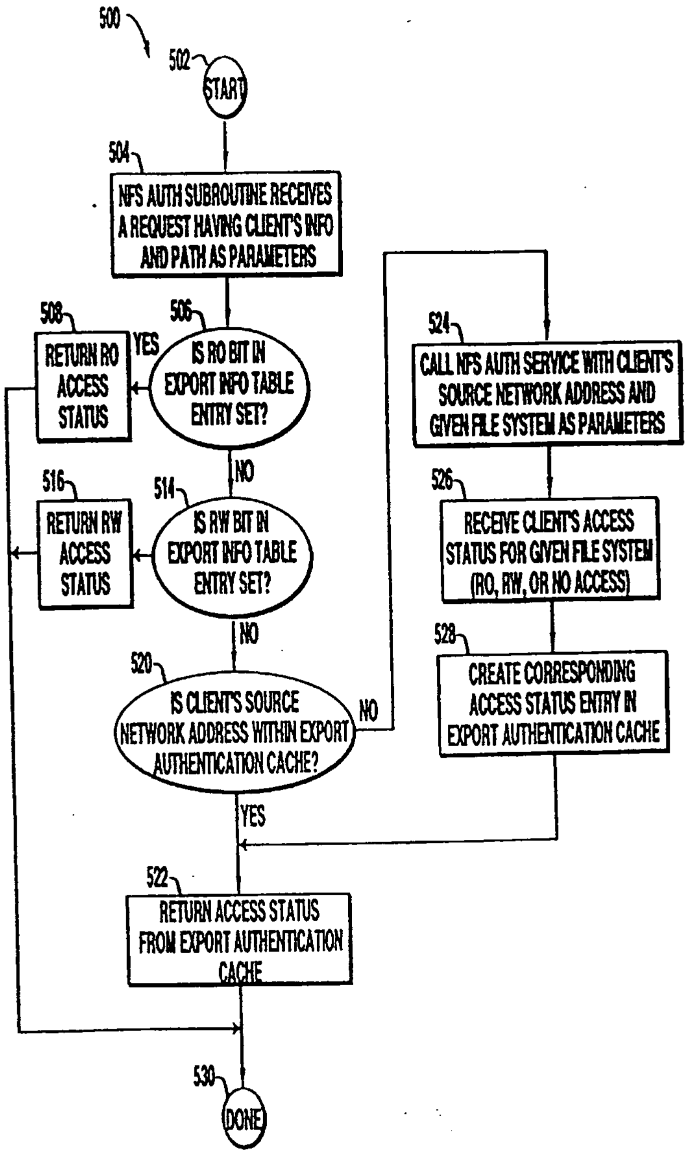

500
502
START
504
NFS AUTH SUBROUTINE RECEIVES A REQUEST HAVING CLIENT'S INFO AND PATH AS PARAMETERS
506
IS RO BIT IN EXPORT INFO TABLE ENTRY SET?
YES
508
RETURN RO ACCESS STATUS
NO
514
IS RW BIT IN EXPORT INFO TABLE ENTRY SET?
516
RETURN RW ACCESS STATUS
NO
520
IS CLIENT'S SOURCE NETWORK ADDRESS WITHIN EXPORT AUTHENTICATION CACHE?
NO
524
CALL NFS AUTH SERVICE WITH CLIENT'S SOURCE NETWORK ADDRESS AND GIVEN FILE SYSTEM AS PARAMETERS
526
RECEIVE CLIENT'S ACCESS STATUS FOR GIVEN FILE SYSTEM (RO, RW, OR NO ACCESS)
528
CREATE CORRESPONDING ACCESS STATUS ENTRY IN EXPORT AUTHENTICATION CACHE
YES
522
RETURN ACCESS STATUS FROM EXPORT AUTHENTICATION CACHE
530
DONE

Fig. 11

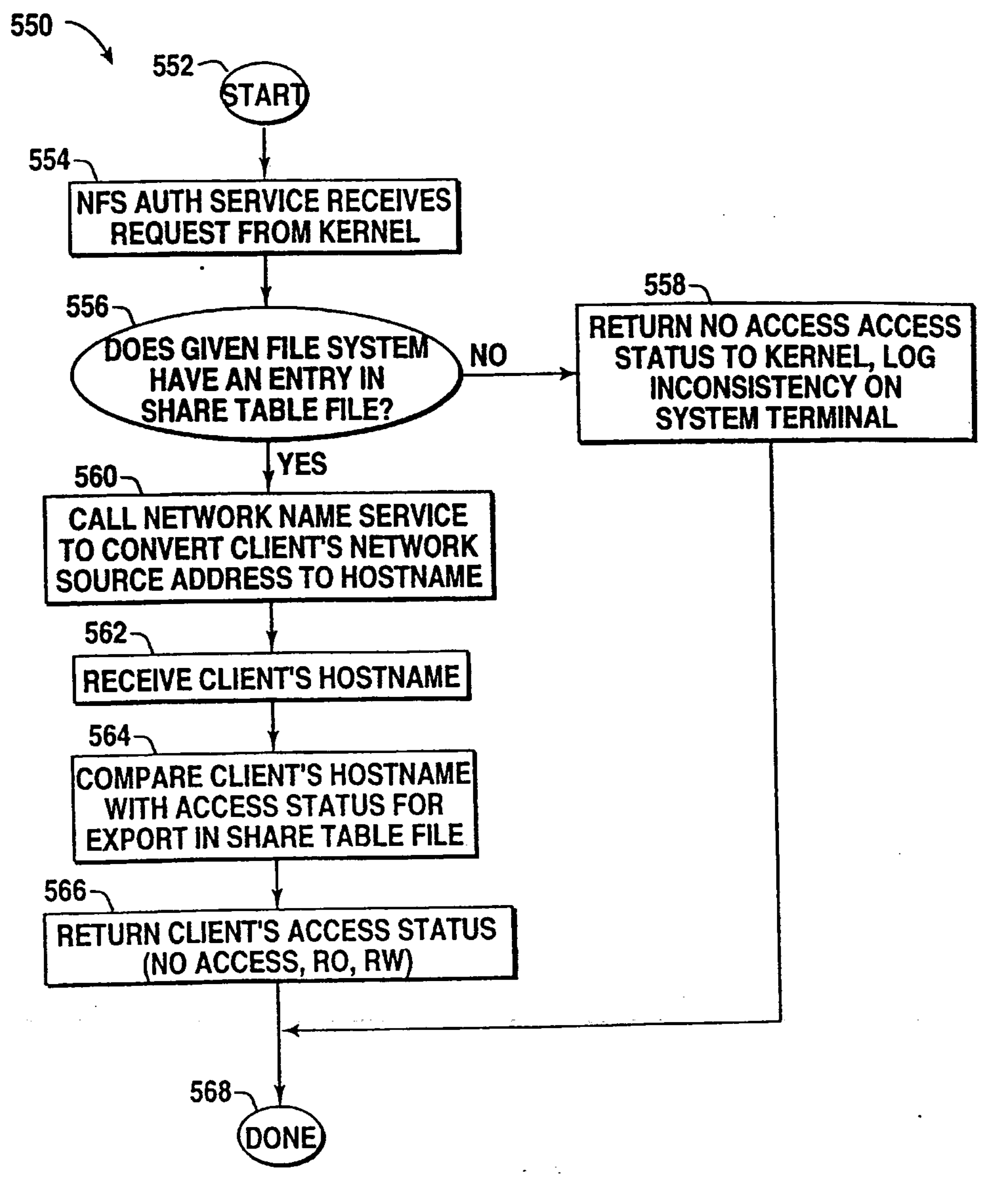

550
552
START
554
NFS AUTH SERVICE RECEIVES REQUEST FROM KERNEL
556
DOES GIVEN FILE SYSTEM HAVE AN ENTRY IN SHARE TABLE FILE?
NO
558
RETURN NO ACCESS ACCESS STATUS TO KERNEL, LOG INCONSISTENCY ON SYSTEM TERMINAL
YES
560
CALL NETWORK NAME SERVICE TO CONVERT CLIENT'S NETWORK SOURCE ADDRESS TO HOSTNAME
562
RECEIVE CLIENT'S HOSTNAME
564
COMPARE CLIENT'S HOSTNAME WITH ACCESS STATUS FOR EXPORT IN SHARE TABLE FILE
566
RETURN CLIENT'S ACCESS STATUS (NO ACCESS, RO, RW)
568
DONE

Fig. 12

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 97 30 0619

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
A	EP 0 398 492 A (IBM) 22 November 1990 * abstract; figure 5 * * page 4, line 56 - page 6, line 22 * * page 12, line 4 - page 13, line 25 * ---	1-30	G06F1/00
A	COMPUTER NETWORKS AND ISDN SYSTEMS, vol. 27, no. 6, 1 April 1995, pages 765-771, XP000498084 LEWONTIN S: "THE DCE WEB TOOLKIT: ENHANCING WWW PROTOCOLS WITH LOWER-LAYER SERVICES" * page 766, left-hand column, line 6 - line 17 * * page 767, left-hand column, line 3 - right-hand column, line 15 * * page 768, left-hand column, line 40 - right-hand column, line 38 * ---	1-30	
A	PROCEEDINGS OF THE SPRING JOINT COMPUTER CONFERENCE, 1972, ATLANTIC CITY, US, pages 417-429, XP002034063 G.S.GRAHAM ET AL: "Protection - Principles and Practice" * the whole document * ---	1-30	TECHNICAL FIELDS SEARCHED (Int.Cl.6) G06F
A	SOFTWARE PRACTICE & EXPERIENCE, vol. 15, no. 9, September 1985, CHICHESTER, GB, pages 889-899, XP002034064 R.J.DAKIN ET AL: "A Large Scale Network Storage Facility" * page 894, left-hand column, line 1 - page 895, left-hand column, line 9 * -----	1-30	

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	30 June 1997	Powell, D

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons
&amp; : member of the same patent family, corresponding document

PATENT ABSTRACTS OF JAPAN

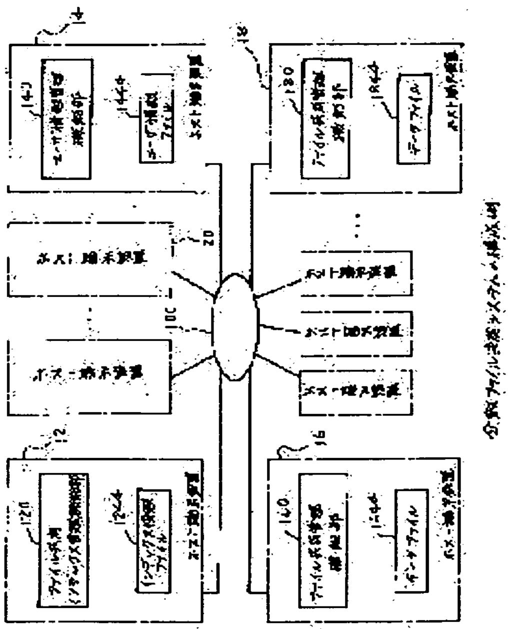
(11)Publication number : 2003-162441
(43)Date of publication of application : 06.06.2003

(51)Int.Cl. G06F 12/00
G06F 15/00
G06F 17/30

(21)Application number : 2001-362287
(22)Date of filing : 28.11.2001
(71)Applicant : OKI ELECTRIC IND CO LTD
(72)Inventor : KOYAMA NORITAKA
WADA KUMIKO

(54) DISTRIBUTED FILE-SHARING SYSTEM AND FILE-ACCESS CONTROL METHOD THEREFOR

(57)Abstract:



PROBLEM TO BE SOLVED: To provide a distributed file-sharing system and its file-access control method, which can realize shortening of retrieval time concerning with access authority and which can realize to increase in efficiency about file operation.

SOLUTION: In this system, a distributed file-sharing system 10 performs management according to access authority, based on index information using an index information file 1244, which is shared by a file-sharing index management function part 120; if host terminal-equipment which a user operate does not have the object of directory information among each host terminal-equipment 16, 18, and so on, the host terminal-equipment can obtain access authority from the file-sharing index function part 120 without accessing each host terminal- equipment, and file-sharing management function part 160, 180, and so on of the host terminal-equipment 16, 18, and so on perform local management, so that the access to each host terminal-equipment 16, 18, and so on, until finishing of process can be made to a minimum.

LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

JP 2003- 162441

CLAIMS

[Claim(s)]
[Claim 1] It has the terminal unit which stores the file which treats data collectively under the directory in which the location of a layered structure is shown, and carries out file management. Each of this terminal unit is connected to a network, and each terminal unit is used as a host terminal unit. Said host terminal unit It includes in index information by making information on rating for accessing the directory and file about a share among these host terminal units into access permission information. The global-area function manager block which manages these all index information is included in [at least one] said network. Said host terminal unit Management of a file including said directory shared between each host terminal unit, and the local file management to this each host terminal unit itself, And the distributed file-sharing system characterized by including the file-sharing function manager block which performs a setup and management of access permission information to the directory used for said file management.
[Claim 2] It is the distributed file-sharing system characterized by managing said global-area function manager block in a system according to claim 1 including said access permission information on said all directories to said index information.
[Claim 3] It is the distributed file-sharing system characterized by said global-area function manager block controlling the reference and updating of said index information in a system according to claim 1 or 2.
[Claim 4] It is the distributed file-sharing system characterized by including user management functional block which this system registers the access permission to said host terminal unit as User Information in a system according to claim 1, 2, or 3, and manages the propriety of an access permission based on authentication of this User Information in said host terminal unit.
[Claim 5] It is the distributed file-sharing system characterized by holding the cache of said access permission information which holds the cache of the file about said share by which said host terminal unit is managed in a system given in claim 1 thru/or any 1 term of 4 for said index information on said global-area function manager block, and the parent directory of this file has.
[Claim 6] It is the distributed file-sharing system characterized by for said file-sharing function manager block on said each host terminal unit notifying renewal of the access permission information carried out locally to said global-area function manager block in a system according to claim 5, and this global-area function manager block notifying renewal of this access permission information further to all other host terminal units that store this access permission information.
[Claim 7] It is the distributed file-sharing system characterized by said access permission information being variable length in a system according to claim 1, 2, 5, or 6.
[Claim 8] It is the distributed file-sharing system characterized by including either the read-out authority over the user name or group name into which said access permission information is registered in the system according to claim 7, write-in authority, said read-out authority and said write-in authority.
[Claim 9] It is the distributed file-sharing system characterized by permitting the creation to said cache of the retrieval and this file to read-out of the file name which said read-out authority has directly under

said directory in a system according to claim 8, or a directory name, and this file, and read-out of this file.
[Claim 10] It is the distributed file-sharing system characterized by said write-in authority permitting a file or new registration of a directory, updating to this file or deletion, and deletion of said directory directly under said directory in a system according to claim 8.
[Claim 11] It is the distributed file-sharing system characterized by including authorization of a file or new registration of a directory, updating to this file or deletion, and deletion of said directory the creation to said cache of the retrieval and this file to read-out of the file name which said read-out authority and said write-in authority have directly under said directory in a system according to claim 8, or a directory name, and this file and authorization of read-out of this file, and directly under said directory.
[Claim 12] In the host terminal unit with which network connection of each of the host terminal unit which stores the file which treats data collectively under the directory in which the location of a layered structure is shown, and carries out file management was carried out The 1st process which outputs an access request to the object of said file management according to the actuation to the information on this directory and either of said files, According to this access request, authority of the user to said object who does access actuation is made into an access permission. When said user has the 2nd process which judges whether there is any access rating over said access request of this user using the access permission managed collectively, and said access permission which satisfies this access rating, When judged with the 3rd process which performs processing corresponding to said access request to said information, and a user without said access rating, The 4th process which outputs the response which avoids the processing corresponding to said access request, The 5th process which processes to this index information by making index information containing said access permission to each of said host terminal unit into an administration object, The file access control approach of the distributed file-sharing system characterized by including the 6th process which outputs the reply signal corresponding to processing of this index information to said operated host terminal unit.
[Claim 13] It is the file access control approach of the distributed file-sharing system characterized by including either the read-out authority over the user name or group name into which said access permission is registered in the approach according to claim 12, write-in authority, said read-out authority and said write-in authority.
[Claim 14] It is the file access control approach of the distributed file-sharing system characterized by permitting the creation to said cache of the retrieval and this file to read-out of the file name which said read-out authority has directly under said directory in an approach according to claim 13, or a directory name, and this file, and read-out of this file.
[Claim 15] It is the file access control approach of the distributed file-sharing system characterized by said write-in authority permitting a file or new registration of a directory, updating to this file or deletion, and deletion of said directory directly under said directory in an approach according to claim 13.
[Claim 16] It is the file access control approach of the distributed file-sharing system characterized by to include authorization of a file or new registration of a directory, updating to this file or deletion, and deletion of said directory the creation to said cache of the retrieval and this file to read-out of the file name which said read-out authority and said write-in authority have directly under said directory in an approach according to claim 13, or a directory name, and this file and authorization of read-out of this file, and directly under said directory.
[Claim 17] In an approach given in claim 12 thru/or any 1 term of 16 the 2nd process The 7th process which judges whether you are the system administrator to whom the user who this accessed manages said whole network when a user accesses from said host terminal unit, The 8th process to which propriety of actuation to said file is made into an access permission, and the access permission of these all files is permitted when this judgment is said system administrator, The 9th process which acquires the access permission to the group to whom this user belongs when said judgment is a different user from said system administrator, The 10th process which judges whether the parent directory information

which expresses the origin for actuation with the host terminal unit to operate is held, The 11th process which acquires the parent directory information which corresponds out of the index information over a file including said directory when said parent directory information judges with an empty condition, The file access control approach of the distributed file-sharing system characterized by including the 12th process which checks the access permission which said parent directory information contains, and performs either authorization of said access permission, and prohibition according to the check result of this access permission.
[Claim 18] The file access control approach of the distributed file-sharing system which progresses to the 12th process including the 13th process said user judges whether you are the owner of said directory between the 11th process and the 12th process to be at the time of the owner of said directory, and is characterized by forbidding said access permission at the times other than this, and ending in an approach according to claim 17.
[Claim 19] It is the file access control approach of the distributed file-sharing system characterized by performing control which maintains the connection relation of the host terminal unit and one to one to which the 5th process sent out said access request during this processing period in the approach given in claim 12 thru/or any 1 term of 18.

[Translation done.]

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[Field of the Invention] This invention is applied to the distributed file system which attains share-ization of the file with which two or more terminal units which distribute a distributed file-sharing system to a network about a distributed file-sharing system and its file access control approach, and are connected are equipped, and is virtually made into one file system, is suitable, treats this distributed file system as an imagination file system, is used for the access control to the directory and the file in this file system, and relates to the suitable file access control approach.
[0002]
[Description of the Prior Art] Usually, the access control list as an OS (OperatingSystem) function is used for the control approach which accesses a file on one computer terminal equipment (henceforth a host terminal unit) used as a host. In the UNIX (trademark) system which is one of the OS's, for every file or directory, it classifies into an owner, an owner group, and other users, the access permission of read-out, writing, and activation is set as each to this classification, and the access control of a file is realized. When using a host terminal unit through a network from a remote terminal, the host terminal unit has judged the access permission to the file and directory which were mentioned above according to a user's account stored in the host terminal unit.
[0003] In addition, in the viewpoint of an access control, when two or more host terminal units share a file server, the approach of carrying out a load distribution (unloading) accompanying this share is proposed. The shared file system in a distributed system given in JP,11-120063,A is attaining improvement in the speed of processing, as the count load of file management is lost and a transfer of file data does not go via a network.
[0004] Shared file management equipment given in JP,9-305470,A supervises the condition of a shared file, and the condition of a file storing means, aims at arrangement of a shared file, or reexamination of multiplexing in the system management section based on a monitor result, performs the migration or multiplexing of a shared file according to reexamination in the file access control section, and is raising employment effectiveness. Moreover, a distributed file system given in JP,8-77054,A Two or more creation and deletion of a division file corresponding to a distributed file are made to perform to two or more server calculating machines on the calculating-machine cluster which carried out network connection of two or more calculating machines by the division file creation section and the division file cutout, respectively. In the distributed file management section, before performing reference/updating demand to a distributed file Reference/updating symmetry information over a division file is delivered to a client computer. By determining the whereabouts of the division file in which the index of the record used for assignment of reference/updating demand in reference/updating demand symmetry section is stored according to this information, even if a processing demand concentrates, a load distribution is carried out effectively.
[0005] Furthermore, as for the file server system of a publication, two or more file servers are installed in JP,6-332782,A side by side through a network, and a file server is accessed at each file enclosure. In

this system a file access demand allocation means The load profile initiation of each file server measured with the load information monitoring means is referred to. When file management which selects the file server which carries out a file access is performed and the selected file server publishes a file access demand through a communications control means to respond for whether being self server or other servers, and to correspond Choose the light file server of a load, and he makes it access at the time of file writing, and is trying for an access request not to concentrate on a specific file server especially. Four proposals mentioned above are proposals based on the viewpoint of an access control, and are completely different techniques from the approach of giving an access permission and controlling access of a file.
[0006]
[Problem(s) to be Solved by the Invention] By the way, two or more host terminal units are distributed and arranged in a network, and the distributed file-sharing system which regarded these like one file system virtually is examined. Generally, in such a distributed file system, it becomes a big problem how unitary management of the access permission in a system is performed. That is, in this distributed file system, one host terminal unit carries out motion control of the access permission management server. If the approach of imitating the file access of the UNIX system mentioned above at this time, and checking all the access permissions of file pass is applied, since both a directory and a file are checked to the judgment of the access permission of a file, a distributed file system may require time amount for retrieval too much, and may not be realistic.
[0007] Since the access permission in an application level is judged to each directory and file of all file pass corresponding to retrieval conditions when the distributed file-sharing system which considered the share nature of a file performs a file search, it is presumed in a distributed file-sharing system that further much retrieval time starts.
[0008] Moreover, since it considers virtually that the distributed file-sharing system was mentioned above with one file system, it becomes possible to carry out the cache of the duplicate of the file of a certain host terminal unit to other host terminal units. Unless a file is deleted, in order to check the access permission about this file, it becomes unnecessary to carry out the file of the host terminal unit which accessed once and was obtained by this cache function through a network. In spite of this situation, in case a host terminal unit judges the access permission of the file to build in, it will ask the server for access permission management through a network. This will make the semantics which carries out the cache of the file to a host terminal unit reduce.
[0009] This invention cancels the fault of such a conventional technique, and it aims at offering the distributed file-sharing system which can attain shortening of the retrieval time in connection with an access permission, and the increase in efficiency of file manipulation, and its file access control approach.
[0010]
[Means for Solving the Problem] In order that this invention may solve an above-mentioned technical problem, it has the terminal unit which stores the file which treats data collectively under the directory in which the location of a layered structure is shown, and carries out file management. Each of this terminal unit is connected to a network, and each terminal unit is used as a host terminal unit. A host terminal unit It includes in index information by making information on rating for accessing the directory and file about a share among these host terminal units into access permission information. The global-area function manager block which manages all these index information is included in [at least one] a network. A host terminal unit Management of a file including the directory shared between each host terminal unit, and the local file management to each of this host terminal unit itself, And it is characterized by including the file-sharing function manager block which performs a setup and management of access permission information to the directory used for file management.
[0011] The distributed file-sharing system of this invention performs management according to an access permission from index information using the index information about the file shared to a global-area function manager block, even if the host terminal unit which a user operates does not have the target directory information, can obtain an access permission from a global-area function manager block

easily, without performing access to each host terminal unit, and performs local management with a file-sharing function manager block.
[0012] Moreover, in order that this invention may solve an above-mentioned technical problem In the host terminal unit with which network connection of each of the host terminal unit which stores the file which treats data collectively under the directory in which the location of a layered structure is shown, and carries out file management was carried out The 1st process which outputs an access request to the object of file management according to the actuation to the information on this directory and either of the files, The 2nd process which judges whether there is any access rating over this user's access request using the access permission which makes an access permission authority of the user to an object who does access actuation, and is collectively managed according to this access request, When a user has the access permission which satisfies this access rating and it is judged with the 3rd process which performs processing corresponding to an access request to information, and a user without access rating, The 4th process which outputs the response which avoids the processing corresponding to an access request, The 5th process which processes to this index information by making index information containing the access permission to each of a host terminal unit into an administration object, It is characterized by including the 6th process outputted to the host terminal unit which operated the reply signal corresponding to processing of this index information.
[0013] In case the file access control approach of the distributed file-sharing system of this invention performs actuation to the information on a directory and either of the files, it can attain shortening of the duration of retrieval compared with the judgment by the file search in each application level by judging by the access permission which is having it set up whether access to the object to this actuation can be performed, and controlling.
[0014]
[Embodiment of the Invention] Next, with reference to an accompanying drawing, one example of the distributed file-sharing system by this invention is explained to a detail.
[0015] This example is the case where the distributed file-sharing system of this invention is applied to the distributed file system 10 which is a virtual file system. Illustration and explanation are omitted about a part without the direct relation to this invention. By the following explanation, a signal is directed with the reference number of the appearing path cord.
[0016] As shown in drawing 1 , two or more host terminal units 12 and 14, 16, 18, 20, and ... are connected to the distributed file system 10 in the network 100. The file-sharing index function manager section 120 and the index information file 1244 are included in the host terminal unit 12. The host terminal unit 14 is equipped with the User Information function manager section 140 and the User Information file 1444. Moreover, the file-sharing function manager section 160, 180 and a data file 1644, and 1844 are arranged in the host terminal unit 16 and 18 the lot every, respectively. Also in the host terminal unit which expressed with the alternate long and short dash line to which the reference mark is not given, either of three kinds corresponds.
[0017] Furthermore, a configuration is explained about each host terminal unit. The host terminal unit 12, 14, 16, 18, and ... are equipped with main frame section 12a, 14a, 16a, 18a, ... and peripheral-device section 12b, 14b, 16b, 18b, and ..., respectively. Main frame section 12a contains CPU(Central Processing Unit) 120a, memory 122, storage 124, and the network connection section 126, as the host terminal unit 12 is shown in drawing 2 . There is a mouse 132 in peripheral-device section 12b as a display 128, a keyboard 130, and a pointing device, and peripheral-device section 12b is connected to main frame section 12a through signal lines 134, 136, and 138.
[0018] Storage 124 is hard disk drive equipment (it is called HDD below Hard Disk Drive unit:). Storage 124 was divided into program storage area 124P and data storage area 124D, and is memorized. The file-sharing index manager 1240 and the actuation interface processing program 1242 are stored in program storage area 124P. The file-sharing index manager 1240 manages the file and the index information file 1244 of all directories by which share assignment was carried out in the host terminal unit 12 linked to a network 100, 14, 16, 18, and ... There is an index information file 1244 in data storage area 124D, and the index information on all files or directories by which share assignment was

carried out is included in it.
[0019] The network connection section 126 is a connection interface with a network 100. The host terminal unit 12 is connected to a network 100 through the network connection section 126, and the distributed file system 10 is building one virtual file system as the whole including other host terminal units 14, 16, 18, and ...
[0020] Main frame section 14a contains CPU 140a, memory 142, storage 144, and the network connection section 146, as the host terminal unit 14 is shown in drawing 3 . There are a display 148, a keyboard 150, and a mouse 152 in peripheral-device section 14b, and peripheral-device section 14b is connected to it through main frame section 14a and signal lines 154, 156, and 158.
[0021] Storage 144 is HDD, and storage 144 was divided into program storage area 144P and data storage area 144D, and it has memorized it. The User Information manager 1440 and the actuation interface processing program 1442 are stored in program storage area 144P. The User Information manager 1440 manages the User Information file 1444 in the host terminal unit 12 linked to a network 100, 14, 16, 18, and ... There is confidential information which specifies a user, such as user ID (IDentifier) and a password, in the User Information file 1444. The User Information file 1444 is stored in data storage area 144D.
[0022] Grouping of the User Information function manager section 140 can be carried out based on the information which defines User Information in the User Information file 1444. Since definition information can be set up variously, a user can belong to two or more groups. The User Information function manager section 140 performs user authentication, when the user who operates it uses an actuation interface. The user ID and confidential information which were mentioned above use a keyboard at the time of a log in, and are inputted into a host terminal unit, and user authentication is performed by collating with User Information in the User Information file 142.
[0023] User Information is user ID, confidential information, etc. more concretely. The User Information function manager section 140 permits actuation only to the congruous valid users as a result of this authentication. At this time, the actuation interface processing program of the corresponding host terminal unit is loaded to memory 122, and actuation is received as actuation interface processing facility section 142a.
[0024] The network connection section 146 is a connection interface with a network 100. The user ID and confidential information from each host terminal unit 12, 16, 18, and ... are supplied through the network connection section 146 at the host terminal unit 12 without the User Information function manager section 140 and the User Information file 142, 16, 18, and ..., and the propriety of an authentication result is returned. A user's access propriety is decided according to this returned result.
[0025] In addition, the file-sharing function manager section later mentioned although not illustrated may be included in the host terminal unit 14.
[0026] Moreover, main frame section 16a contains CPU 160a, memory 162, storage 164, and the network connection section 166, as the host terminal unit 16 is shown in drawing 4 . There are a display 168, a keyboard 170, and a mouse 172 in peripheral-device section 16b, and peripheral-device section 16b is connected to it through main frame section 16a and signal lines 174, 176, and 178.
[0027] By HDD, storage 164 was divided into program storage area 164P and data storage area 164D, and has memorized storage 164. The file-sharing manager 1640 and the actuation interface processing program 1642 are stored in program storage area 164P. The file-sharing manager 1640 is managed according to the data file 1644 by which share assignment was separately carried out in the host terminal unit 16 linked to a network 100, 18, and ..., and the processing to 1844. Directory information and file information are stored in the data file 1644 of data storage area 164D.
[0028] The file-sharing function manager section 160 can aim at cooperation with other host terminal units 12, 14, 18, the file-sharing index function manager section 120 of ..., the User Information function manager section 140 and the file-sharing function manager section 180, and ... while realizing a file and the actuation function of a directory locally. A file and the actuation function of a directory are an add function, a retrieval function, an acquisition function, a read-out function, an updating function, and the Delete function, as the latter part explains.

[0029] In a distributed file system 10, the host terminal unit 16, the same host terminal unit 18 of a configuration, and ... distribute, and it is arranged at it. The host terminal unit 18 has the file-sharing function manager section 180.
[0030] Next, the information on data storage area 164D treated by the distributed file system 10, and 184D and 124D is explained. The host terminal unit 16 contains a data file 1644 in data storage area 164D. A data file 1644 contains directory information P and file 164A (= "/P/A") to a directory "/P", as shown in drawing 5 (a). Here, file 164A holds file information A and file 164a as one file. A directory "/P" and file 164A are operated through actuation interface processing facility section 162a, and are created and registered by the function of the file-sharing function manager section 160.
[0031] However, file 164a is not necessarily the file of the file pass "/P/A" in OS used with the host terminal unit 16. When the file which is under management of the file-sharing function manager section 160 from actuation interface processing facility section 162a is displayed, the tree structure of file 164A is shown to a directory "/P" like the local view of drawing 5 (b).
[0032] Similarly, the host terminal unit 18 containing the file-sharing function manager section 180 has directory information Q and R in the data storage area of drawing 6 (a), file information B and file 184b, and file 184C are file information C and file 184c, and file 184B associates it, respectively. More specifically, the file is managed in the relation of a directory "/Q", "/Q/R" and a file "/Q/184B", and "Q/R/184C." The directory and file which were mentioned above are operated through actuation interface processing facility section 182a, respectively, and are created and registered by the function of the file-sharing function manager section 180. It is not necessarily the file of the file pass "/Q/184B" in OS which uses file 184b and 184c with the host terminal unit 18 also in this case, and "/Q/R/184C." When the file which is under management of the file-sharing function manager section 180 from actuation interface processing facility section 182a is displayed, the tree structure of file 184C is shown to file 184B and a directory "/Q/R" like the local view of drawing 6 (b) to a directory "/Q."
[0033] Moreover, the host terminal unit 12 carries out unitary management of directory information [of a distributed file system 10] P, Q, R and file information A, B, and all C in the file-sharing index function manager section 120, as shown in drawing 7 (a). It is stored in index information, the call, and the index information file 1244 to these [all]. If the index information file 1244 is indicated by the global view by management of the file-sharing index function manager section 120, the index information file 1244 containing directory information and file information is "/directory P"-file information A, as shown in drawing 7 (b), "/directory Q"-file information B, And it is expressed with the imagination tree structure of directory "/Q/R"-file information C.
[0034] Next, the directory information and file information which were mentioned above are explained. Directory information has a key 40, a directory name 42, the directory refix date 44, the owner 46, the cancellation day 48, the open day 50, and the access permission 52, as shown in drawing 8 . A key 40 is a value showing a directory. A directory name 42 shows the virtual directory path name in a distributed file system 10. The directory refix date 44 is the information on a day that this directory was updated at the end. An owner 46 expresses an owner name by making the implementer of a directory into an owner. Directory information can update only system administrator admin with this owner.
[0035] Moreover, there are a cancellation day 48 and a open day 50 as a day which specifies the outside for retrieval. The cancellation day 48 is an expiration date end date the target [retrieval] as which outside for retrieval is set when it passes over this day, and the open day 50 is an expiration date opening day for [which makes the day before this the outside for retrieval] retrieval. An access permission 52 contains an access control list.
[0036] An access control list makes a user name or a group name, and an access permission correspond to drawing 9 so that it may be shown. A user name and a group name are names registered into the User Information file 1444 by the User Information Management Department function part 140. There are read-out (henceforth r) authority, write-in (henceforth w) authority, and read-out write-in authority (henceforth rw) authority in an access permission, and any one authority makes it correspond to a user name or a group name among these three. The access control list is made into variable length. The latter part explains an access permission further.

[0037] File information includes a key 60, a file name 62, the file updating day 64, the owner 66, the cancellation day 68, the open day 70, and the condition 72 of a file, as shown in drawing 10. Since the item of the same name as directory information is the same definition in file information, explanation is omitted. Here, a file name 62 is a virtual-file pathname, and an owner 66 is a registrant of a file. Moreover, the file is the identifier which distinguishes original or a cache, i.e., the reproduced thing, in the condition 72 of a file.
[0038] Here, the file-sharing management at the time of gaining, when the host terminal unit 16 carries out the cache of the file 184B from the host terminal unit 18 (duplicate) is shown in drawing 11. With this acquisition, as shown in drawing 11 (a), directory information Q and file 184B are added to data storage area 164D of the host terminal unit 16. File 184B contains file information B and file 184b. The slash is given to the information by which the cache was carried out. Therefore, when a user makes the file display under management of the file-sharing function manager section 160 perform from actuation interface function part 162a, file 184B is visible to a directory "/P" by file 164A and the directory "/Q" with a local view.
[0039] However, file 184B also shows that it is a cache (duplicate) as shown by the slash.
[0040] Next, the access permission in directory information is explained. For example, r authority in Directory P is the authority over the directory P from the file-sharing function manager section of the host terminal unit of arbitration, and is the read-out authority of the directory name of a directory P directly under the authority of retrieval / acquisition / read-out processing of as opposed to [as opposed to / to the 1st / the read-out authority of the file name of a directory P directly under] the file of a directory P directly under to the 2nd, and the 3rd. Meaning that the acquisition processing mentioned above creates the cache of the read file, read-out processing means mere read-out.
[0041] Moreover, w authority in Directory P is the authority over the directory P from the file-sharing function manager section of the host terminal unit of arbitration, and is updating/deletion authority of the file of a directory P directly under the creation authority of a directory new directly under directory P to the 1st, and the 2nd in the registration authority of a file new directly under directory P to the deletion authority of the directory of a directory P directly under, and the 3rd, and the 4th. However, renewal of a cache file cannot be performed.
[0042] In the distributed file system 10, the system administrator account admin exists and Account admin has rw authority to all directories. Moreover, the owner of a directory has rw authority to the directory to own.
[0043] Next, the judgment procedure of an access permission is explained. A user has got the propriety of access actuation in response to the judgment of an access permission, before performing directory actuation and file manipulation. As shown in drawing 12, an actuation user judges that he is a system administrator (step S10). User ID actually judges this judgment by whether it is the system administrator account admin. When user ID is in agreement with the system administrator account admin (YES), access of all directories and files is permitted to an actuation user (step S12). The access permission at this time is rw authority. It shifts to termination after this judgment.
[0044] Moreover, when an actuation user is not the system administrator account admin, all the groups to whom an actuation user belongs in (NO) and a distributed file system 10 are acquired (step S14). When this group information logs in to the host terminal unit to be used, it can operate the User Information function manager section 140 of the host terminal unit 14, and it can be acquired from the User Information file 1444.
[0045] Next, it judges whether the directory and the parent directory information on a file which are made applicable to actuation are on the host terminal unit currently operated (step S16). When parent directory information is not held, the host terminal unit under (NO) and actuation acquires parent directory information from the index information file 1244 through the file-sharing index function manager section 120 of the host terminal unit 12 (step S18).
[0046] Moreover, after acquiring the case where parent directory information is held, and parent directory information, an access permission is checked with reference to the access control list contained in parent directory information (step S20).

[0047] Next, it judges whether the entry which shows at least one authorization exists in the access permission obtained by the check (step S22). When an authorization entry exists (YES), it enables it for the host terminal unit under actuation to perform processing according to authorization (authorization: step S24). Moreover, when an authorization entry does not exist, access to (NO), a directory, and a file is forbidden (step S26). It shifts to termination through these processings.
[0048] The judgment of an access permission has not only the procedure mentioned above but the judgment processing performed when updating the drawing 13 directory information. To the process in which the same processing as the verification procedure of the access permission of drawing 12 is performed, the reference mark attached with the flow chart of drawing 12 is given, and explanation is sharply omitted in it. By the judgment (step S10) of being a system administrator, when an actuation user is a system administrator, all are permitted and user authentication is ended. When judged with actuation users other than this not being system administrators, it judges whether the parent directory information (NO) and for actuation is held (step S16). When this parent directory is not held, (NO) file-sharing index function manager section 120 is operated, and it acquires from the index information file 1244.
[0049] Next, with reference to the directory information currently held or the acquired directory information, an actuation user checks that he is the owner of a directory (step S28). When the owner of directory information is compared with an actuation user's user ID and it is in agreement (YES), renewal of informational is permitted (step S24). Moreover, when this comparison is an inequality, renewal of (NO) and information is forbidden, and it is made disapproval (step S26). Thus, in renewal of information, user authentication is performed, and the existence of updating rating is judged and it ends.
[0050] The sequence of the directory actuation in a distributed file system 10 and file manipulation is explained performing a judgment and user authentication of such an access permission. A directory and the sequence about a file are as being fundamentally shown in drawing 14. This fundamental sequence is creation of a directory, the renewal of directory information, and deletion of a directory.
[0051] In the sequence of drawing 14, the expedient top of explanation and the host terminal unit 16 are used. The operator guidance signal 200 over a directory is supplied to actuation interface processing facility section 162a by the key stroke at time of day T10. Actuation interface processing facility section 162a sends out the directions demand signal 202 to the file-sharing function manager section 160 at time of day T12.
[0052] In the file-sharing function manager section 160, judgment processing of the access permission to a directory is started at time of day T14. At time of day T16, the file-sharing function manager section 160 sends out the information acquisition demand signal 204 of parent directory information to the file-sharing index function manager section 120 of the host terminal unit 12, when there is no directory information. The file-sharing index function manager section 120 starts retrieval of the parent directory information demanded based on the index information file 1244 which is carrying out unitary management at time of day T18. The file-sharing index function manager section 120 sends out the parent directory information acquired by retrieval to the file-sharing function manager section 160 as an acquisition information signal 206 at time of day T20.
[0053] The file-sharing function manager section 160 checks an access permission based on the parent directory information supplied at time of day T22. When it judges with the file-sharing function manager section 160 not having the access permission obtained according to a check, the reply signal 208 which shows that there is no authority at time of day T24 is sent out to actuation interface processing facility section 162a. Furthermore, actuation interface processing facility section 162a outputs the reply signal 210 which shows that there is no authority to the display 168 which a user uses. Thereby, a user gets to know that there is no access permission.
[0054] Moreover, when parent directory information exists in the data file 1644 under management in the file-sharing function manager section 160 in judgment processing of an access permission, an access permission is immediately checked at time of day T16. When it is judged that there is no access permission, it progresses to the response processing which performs subsequent processings at time of day T24, and the time of day T24 mentioned above and response processing of T26 are performed

further.
[0055] On the other hand, when there is an access permission as a result of the check judging of an access permission, it progresses to demand processing at time of day T28 from the phase of time of day T16 or time of day T22. At this time, the file-sharing function manager section 160 performs each processing according to a directions demand, and notifies it to the file-sharing index function manager section 120 by making into the notice signal 212 of information directory information processed at time of day T28. A distributed file system 10 has a possibility that mismatching may occur, even if an actuation demand is given from two or more host terminal units to coincidence by the file-sharing index function manager section 120 through a network 100.
[0056] In order to prevent generating of this mismatching, the host terminal unit 120 performs connection control of one to one which communicates only with one host terminal unit. If it sees in another viewpoint, exclusive control is performed so that the file-sharing index function manager section 120 cannot communicate except the host terminal unit under connection. The processing to an administration object is completed from time of day T30, and this exclusive control is continued till the time of day T32 when the file-sharing index function manager section 120 sends out a reply signal 214 to the file-sharing function manager section 160.
[0057] The file-sharing function manager section 160 outputs the reply signal 216 which shows that demand processing was completed at time of day T34 to actuation interface processing facility section 162a. Furthermore, actuation interface processing facility section 162a sends out a reply signal 218 to a display 168 at time of day T36. Thereby, an actuation user gets to know that desired processing was completed.
[0058] A more concrete example is given and it explains briefly. In directory creation, in a distributed file system 10, the host terminal unit 16 operates directory creation through actuation interface processing facility section 162a from the host terminal unit 16 (time of day T12). The file-sharing function manager section 160 starts judgment processing of whether an actuation user has w authority among access permissions (time of day T14).
[0059] When w authority is checked, the file-sharing function manager section 160 will create directory information by time of day T28 based on the input data supplied through actuation interface processing facility section 162a from the keyboard 170. The file-sharing function manager section 160 notifies the directory information created at time of day T28 to the file-sharing index function manager section 120. Between time of day T30-T32, performing exclusive control mentioned above, the file-sharing index function manager section 120 is added to the index information file 1244 by making notified directory information into an administration object, and updates index information.
[0060] When there was no w authority and it is checked at time of day T16 or time of day T22, the file-sharing function manager section 160 progresses to processing of time of day T24, and performs future response processings.
[0061] When updating directory information, the host terminal unit 16 performs directory information update operation through actuation interface processing facility section 162a. In response to this actuation, the file-sharing function manager section 160 performs access judging processing of user authentication according to whether directory information is owned at either time of day T16 and the time of day T22. When [which an actuation user calls the owner or system administrator of a directory] judged, access to which the host terminal unit 16 updates directory information is permitted.
[0062] The file-sharing function manager section 160 creates the directory information updated based on the input data supplied through actuation interface processing facility section 162a by time of day T28 from the keyboard 170. The file-sharing function manager section 160 notifies the directory information created at time of day T28 to the file-sharing index function manager section 120. Between time of day T30-T32, performing exclusive control mentioned above, the file-sharing index function manager section 120 is added to the index information file 1244 by making notified directory information into an administration object, and updates index information.
[0063] By the way, since the file-sharing index function manager section 120 knows whether the directory updated to which host terminal unit exists, it notifies the updated directory information. This

notice is called notice of an event. The notice of an event is not limited to updating and performed also in the directory deletion, file updating, and file deletion which are mentioned later. From the procedure shown below being the same also in directory deletion, file updating, and file deletion, the explanation in the latter part is simplified about three above-mentioned processings, and it explains paying attention to a different point.
[0064] Suppose that the host terminal unit 18 and 20 have a directory applicable to this notice of an event. In this case, as shown in drawing 15, the file-sharing index function manager section 120 notifies time of day T40 and the directory information updated by T44 as the notice signal 220 of an event, and 222 to the host terminal unit 18 and 20, respectively. The file-sharing function manager section 180 will update the directory information in a data file 1844 by time of day T48 in response to the notice of an event at time of day T42. Moreover, file-sharing function manager section 20a will update data file (not shown) directory information by time of day T50 in response to the notice of an event at time of day T46.
[0065] The file-sharing function manager section 180 notifies the directory information updated in the file-sharing index function manager section 120 to time of day T48 as a notice signal 224 of updating. Similarly, file-sharing function manager section 20a notifies the directory information updated at time of day T50 as a notice signal 226 of updating. During time of day T52-T54 is controlled exclusively, and the file-sharing index function manager section 120 updates the host terminal unit 18 and the index information on the index information file 1244 of 20 the supplied notice signal 224 of updating, and based on 226. The file-sharing index function manager section 120 outputs a reply signal 228 and 230 to the host terminal unit 18 and 20 in time of day T54 and T56, respectively. Thus, renewal of the directory information corresponding to the notice of an event is performed by the processing by time of day T40-T56.
[0066] Although a series of notices of an event mentioned above, an update process, renewal of index information, and a response are performed in consideration of sequence to the host terminal unit 18 and 20 here, if a series of procedures to each host terminal unit are observed, even if the sequence of a host terminal unit is in random order, it is good.
[0067] When deleting directory information, the host terminal unit 16 performs deletion actuation of a directory through actuation interface processing facility section 162a (time of day T12). The file-sharing function manager section 160 starts judgment processing of whether an actuation user has w authority among access permissions (time of day T14). When w authority is checked, the file-sharing function manager section 160 will delete the directory information over the directory of the location pointed out on the display 168 through the input data or the mouse 172 supplied through actuation interface processing facility section 162a from the keyboard 170 by time of day T28.
[0068] The file-sharing function manager section 160 notifies the directory information deleted at time of day T28 to the file-sharing index function manager section 120. The file-sharing index function manager section 120 is deleted from the index information file 1244 by making notified directory information into an administration object between time of day T30-T32, performing exclusive control mentioned above. The file-sharing index function manager section 120 also performs the notice of an event of directory deletion.
[0069] It is carried out by the sequence which also showed the sequence about a file fundamentally to drawing 14. The file-sharing function manager section 160 performs registration of a file, retrieval and acquisition (read-out), updating, and deletion as processing. When registering a file, register operation of a file is performed through actuation interface processing facility section 162a. In the file-sharing function manager section 160, judgment processing of the actuation user to w authority is performed among access permissions. When there is no access permission, the file-sharing function manager section 160 outputs the reply signal 208 which shows that there is no authority at time of day T24 to actuation interface processing facility section 162a. Moreover, when it judges with there being w authority, the file-sharing function manager section 160 registers a file into a data file 1644. Under the present circumstances, the file-sharing function manager section 160 makes an administration object the file which creates and registers file information. The file-sharing function manager section 160 is

notified to the file-sharing index function manager section 120 by making into the notice signal 212 of information file information created at time of day T28.
[0070] The file-sharing index function manager section 120 controls exclusively, is added to the index information file 1244 by making supplied file information into an administration object, and updates index information. The file-sharing index function manager section 120 outputs a reply signal 214 to the file-sharing function manager section 160 at time of day T32 after this updating.
[0071] When updating a file, update operation of a file is performed through actuation interface processing facility section 162a. In the file-sharing function manager section 160, judgment processing of the actuation user to w authority is performed among access permissions. When there is no access permission, the file-sharing function manager section 160 outputs the reply signal 208 which shows that there is no authority at time of day T24 to actuation interface processing facility section 162a.
[0072] Moreover, when it judges with there being w authority, the file-sharing function manager section 160 updates a file to a data file 1644. Under the present circumstances, the file-sharing function manager section 160 creates the file information according to updating, and makes an administration object the file which registers. Here, the file which can update the file-sharing function manager section 160 is only an original file. The cache file created by the duplicate in the file-sharing function manager section 160 cannot be updated. Renewal of a cache file is performed by outputting an acquisition demand to the target host terminal unit in response to the notice of an event from the file-sharing index function manager section 120 so that it may mention later.
[0073] The file-sharing function manager section 160 is notified to the file-sharing index function manager section 120 by making into the notice signal 212 of information file information created at time of day T28. The file-sharing index function manager section 120 controls exclusively, and updates the index information on the index information file 1244 by making supplied file information into an administration object. The file-sharing index function manager section 120 outputs a reply signal 214 to the file-sharing function manager section 160 at time of day T32 after this updating.
[0074] By the way, the file-sharing index function manager section 120 is carrying out unitary management of in which host terminal unit the updated cache file is included. Thereby, the file-sharing index function manager section 120 gives an event notice to the host terminal unit 18 for updating, and 20 by the sequence mentioned above.
[0075] In addition, in this notice processing of an event, the sequence which is not included in the sequence of drawing 15 is performed. In this sequence, the file-sharing function manager section 180 and 20a output the acquisition demand of an update file to the file-sharing function manager section 160. The file-sharing function manager section 160 outputs the file information and the update file which were updated according to the acquisition demand supplied to the file-sharing function manager section 180 and 20a. The file-sharing function manager section 180 and 20a update a cache file according to the file information and the update file which are supplied and which were updated. The file-sharing function manager section 180 and 20a notify the notice signal 224 of updating, and the file information updated as 226 to the file-sharing index function manager section 120, respectively. The file-sharing index function manager section 120 controls exclusively, and updates the index information on the index information file 1244 by the file-sharing function manager section 180 and the file information supplied from 20a. The file-sharing index function manager section 120 outputs a reply signal 228 and 230 in time of day T54 and T56 after this updating, respectively.
[0076] Next, when deleting a file, deletion actuation of a file is performed through actuation interface processing facility section 162a. In the file-sharing function manager section 160, judgment processing of the actuation user to w authority is performed among access permissions. When there is no access permission, the file-sharing function manager section 160 outputs the reply signal 208 which shows that there is no authority at time of day T24 to actuation interface processing facility section 162a. Moreover, when it judges with there being w authority, the file-sharing function manager section 160 deletes a file to a data file 1644. The file management function part 160 is notified to the file-sharing index function manager section 120 by making eliminated file information into the notice signal 212 of information. Deletion of the cache file in the host terminal unit 16 is to this notice phase.

[0077] When deleting the original file in the host terminal unit 16, further, the file-sharing index function manager section 120 controls exclusively, and deletes the file information supplied from the file-sharing function manager section 160 from the index information file 1244. Since the file-sharing index function manager section 120 carried out unitary management and knows the host terminal unit with which the cache file reproduced about the eliminated file exists, it gives an event notice like updating mentioned above, and deletes the eliminated file information of a file and the eliminated file in the file-sharing function manager section 180 and 20a. The file-sharing index function manager section 120 deletes the host terminal unit 18 and the file information of 20 from the index information file 1244 in response to the notice of the file-sharing function manager section 180 and the file information deleted from 20a.
[0078] It explains referring to drawing 16 about the case where retrieval of a file and acquisition (read-out) are performed, finally. The host terminal unit 16 inputs the operator guidance signal 200 which directs a file search to actuation interface processing facility section 162a (time of day T10). Actuation interface processing facility section 162a outputs the directions demand signal 202 to the file-sharing function manager section 160 as a file search demand (time of day T12). The file-sharing function manager section 160 starts judgment processing of whether an actuation user has r authority among access permissions (time of day T14). Here, the file-sharing function manager section 160 investigates the existence of directory information to an actuation user.
[0079] When there is no directory information of an actuation user in the file-sharing function manager section 160, the file-sharing function manager section 160 outputs the information acquisition demand signal 204 to the file-sharing index function manager section 120 (time of day T16). The access permission list of actuation users is searched with the file-sharing index information management function part 120 from the index information file 1244 after time of day T18. The file-sharing index function manager section 120 acquires the corresponding directory information which included the access permission list between time of day T18-T20 from the index information file 1244. The acquired directory information is supplied to the file-sharing function manager section 160 as an acquisition information signal 206 from the file-sharing index function manager section 120 (time of day T22).
[0080] Moreover, when the file-sharing function manager section 160 has an actuation user's directory information, as for the file-sharing function manager section 160, an access permission performs judgment processing of being r authority at time of day T16. Therefore, the file-sharing function manager section 160 performs judgment processing of whether there is any r authority among access permissions from the directory information supplied at either time of day T16 and the time of day T22. When it judges with there being no r authority, the file-sharing function manager section 160 outputs the reply signal 208 which shows that there is no authority at time of day T24 to actuation interface processing facility section 162a. Actuation interface processing facility section 162a outputs a reply signal 210 to a display 168 at time of day T26 in response to a reply signal 208.
[0081] When r authority is checked, the file-sharing function manager section 160 will be notified to the file-sharing index function manager section 120 by time of day T28 by making into the notice signal 212 of information the retrieval data supplied through actuation interface processing facility section 162a from the keyboard 170. The file-sharing index function manager section 120 searches file information managed by the index information file 1244 to the notified retrieval data between time of day T30-T32, performing exclusive control mentioned above. The file-sharing index function manager section 120 searches that the file for which it asks exists in the host terminal unit 18 by collating of the file information which is in agreement with retrieval data.
[0082] The file-sharing index function manager section 120 is outputted to the file-sharing function manager section 160 by making a retrieval result into a reply signal 212 at time of day T32. The file-sharing function manager section 160 outputs a reply signal 216 to actuation interface processing facility section 162a at time of day T34, and actuation interface processing facility section 162a outputs a reply signal 218 to time of day T36. The host terminal unit 16 can obtain a file search result more quickly than before, and can be made to display it on a display 168 in the distributed system which this built virtually.

[0083] The case where file acquisition is carried out to file search processing continuing is explained. In this case, the host terminal unit 16 inputs directions of file acquisition through actuation interface processing facility section 162a at time of day T60. Actuation interface processing facility section 162a outputs a file acquisition demand to the file-sharing function manager section 160 at time of day T62. The file-sharing function manager section 160 which received this demand at time of day T64 outputs the information acquisition demand signal 204 as a file acquisition demand to the host terminal unit 18 obtained by the above-mentioned file search.
[0084] Here, the file-sharing function manager section 160 investigates whether a directory unprecedented in a data file 1644 is included from the file pass which a file search result shows with the view of a local view, before carrying out a file acquisition demand. The file-sharing function manager section 160 also requires acquisition of the directory information corresponding to this directory of the file-sharing function manager section 180, when the directory which was not in a local view is included in the file search result.
[0085] The file-sharing function manager section 180 is outputted to the file-sharing function manager section 160 by making into the acquisition information signal 206 the file information and the file which correspond at time of day T70 according to the demand received at time of day T68. When the acquisition demand of the directory information which the file-sharing function manager section 160 mentioned above is also doubled and advanced, directory information also includes and sends out the file-sharing function manager section 180 to the acquisition information signal 206.
[0086] The file-sharing function manager section 160 carries out the cache of the supplied information (time of day T72). And the file-sharing function manager section 160 notifies the notice signal 212 of information as information acquired in the file-sharing index function manager section 120 to time of day T74. When directory information is also acquired, it cannot be overemphasized that directory information is also included in above-mentioned information. The file-sharing index function manager section 120 updates the acquired information new to the host terminal unit 16 as index information in the index information file 1244 between time of day T76-T78, controlling exclusively. The file-sharing index function manager section 120, the file management function part 160, and actuation interface processing facility section 162a output time of day T78, T80, the reply signal 214 that shows the completion of updating by T82, 216, and 218, respectively.
[0087] In processing of a file search and file acquisition, since processing is performed continuously, judgment processing of an access permission is performed only once. When not continuing processing but operating each processing separately, in each actuation, this judgment processing is performed by a unit of 1 time, respectively. Moreover, it cannot be overemphasized that access judging processing, access-control processing, and exclusive control processing are performed as mentioned above in the distributed file system 10 also in mere file read-out actuation.
[0088] Thus, the duration which processing takes compared with the case where judge an access permission in each application level simply, and creation, updating, deletion, retrieval, and acquisition are performed can be sharply shortened by treating as a distributed file virtually according to each case, and centralized-control-processing based on index information.
[0089] By constituting as mentioned above, a distributed file system 10 Management according to an access permission is performed from index information using the index information about the file shared in the file-sharing index function manager section 120. Even if the host terminal unit which a user operates among each host terminal unit 16, 18, and ... does not have the target directory information An access permission can be obtained from the file-sharing index function manager section 120, without accessing each host terminal unit. By performing management local at each host terminal unit 16, 18, the file-sharing function manager section 160 of ..., 180, and ... Since access to each host terminal unit 16 performed by the completion of processing, 18, and ... can be made into the minimum, the duration of processing can be finished in a short time rather than before. Therefore, a distributed file system 10 can offer a user-friendly system.
[0090] Moreover, access to the file-sharing index function manager section 120 can be lost with judgment processing of the access permission to the file which carried out the cache, the cache file

which each host terminal unit has can be used effectively, and the processing to a distributed file can be made to perform efficiently by the file-sharing index function manager section's 120 carrying out unitary management, holding the directory information which each of a host terminal unit has, and using this directory information for judgment processing of an access permission.
[0091] Since renewal of an access permission is performed to all the host terminal units that are carrying out the cache of the changed directory when an access permission is changed especially, the coordination of the access permission in a system can be maintained.
[0092]
[Effect of the Invention] Thus, according to the distributed file-sharing system and its file access control approach of this invention Even if the host terminal unit which performs management according to an access permission from index information using the index information about the file shared to a global-area function manager block, and a user operates does not have the target directory information By being able to obtain an access permission from a global-area function manager block, and performing local management with a file-sharing function manager block, without performing access to each host terminal unit Since access to each host terminal unit performed by the completion of processing can be made into the minimum, the duration of processing can be finished in a short time rather than before. This system can be offered as a user-friendly system.
[0093] Moreover, access to a global-area function manager block is lost with judgment processing of the access permission to the file which carried out the cache, the cache file which it has can be used effectively and each host terminal unit can make the processing to a distributed file perform efficiently by a global-area function manager block carrying out unitary management, holding the directory information which each of a host terminal unit has, and using this directory information for judgment processing of an access permission.

[Translation done.]

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報**（A）　(11)特許出願公開番号
特開2003－162441
（P2003－162441A）
(43)公開日　平成15年6月6日(2003. 6. 6)

(51)Int.Cl.⁷		識別記号	ＦＩ			テーマコード゛(参考)
Ｇ０６Ｆ	12/00	５４５	Ｇ０６Ｆ	12/00	５４５Ａ	５Ｂ０７５
		５３７			５３７Ａ	５Ｂ０８２
	15/00	３１０		15/00	３１０Ｕ	５Ｂ０８５
		３３０			３３０Ｄ	
	17/30	１１０		17/30	１１０Ｃ	

審査請求　未請求　請求項の数19　ＯＬ　（全 19 頁）　最終頁に続く

(21)出願番号　特願2001－362287(P2001－362287)

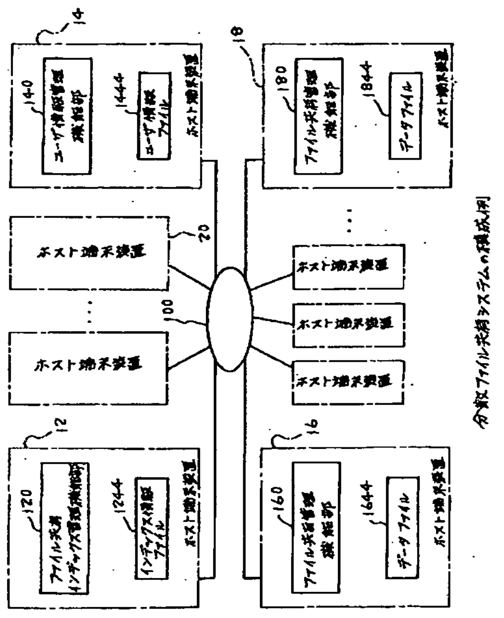
(22)出願日　平成13年11月28日(2001. 11. 28)

(71)出願人　000000295
沖電気工業株式会社
東京都港区虎ノ門１丁目７番12号
(72)発明者　小山　法孝
東京都港区虎ノ門１丁目７番12号　沖電気工業株式会社内
(72)発明者　和田　久美子
東京都港区虎ノ門１丁目７番12号　沖電気工業株式会社内
(74)代理人　100079991
弁理士　香取　孝雄

最終頁に続く

(54)【発明の名称】　分散ファイル共有システムおよびそのファイルアクセス制御方法

(57)【要約】
【課題】　アクセス権限に関わる検索時間の短縮化およびファイル操作の効率化を図ることができる分散ファイル共有システムおよびそのファイルアクセス制御方法の提供。
【解決手段】　分散ファイルシステム10は、ファイル共有インデックス管理機能部120で共有するインデックス情報ファイル1244を用いてインデックス情報からアクセス権限に応じた管理を行い、各ホスト端末装置16, 18, ・・・のうち、ユーザが操作するホスト端末装置が対象のディレクトリ情報を持っていなくても、各ホスト端末装置にアクセスすることなく、ファイル共有インデックス管理機能部120からアクセス権限を得ることができ、各ホスト端末装置16, 18, ・・・のファイル共有管理機能部160, 180, ・・・で局所的な管理を行い、処理完了までの各ホスト端末装置16, 18, ・・・へのアクセスを最小限にすることができる。



【特許請求の範囲】
【請求項１】　データをまとめて扱うファイルを階層構造の位置を示すディレクトリの下に格納してファイル管理する端末装置を備え、該端末装置のそれぞれをネットワークに接続させ、各端末装置をホスト端末装置とし、
前記ホスト端末装置は、これらのホスト端末装置のうち、共有に関するディレクトリおよびファイルにアクセスする資格の情報をアクセス権限情報としてインデックス情報に含め、該インデックス情報すべてを管理する大域管理機能ブロックを前記ネットワーク内に少なくとも一つ含み、
前記ホスト端末装置は、各ホスト端末装置で共有する前記ディレクトリを含めたファイルの管理および該各ホスト端末装置自体に対する局所的なファイル管理、ならびに前記ファイル管理に用いるディレクトリに対するアクセス権限情報の設定および管理を行うファイル共有管理機能ブロックを含むことを特徴とする分散ファイル共有システム。
【請求項２】　請求項１に記載のシステムにおいて、前記大域管理機能ブロックは、前記インデックス情報に前記ディレクトリすべての前記アクセス権限情報を含めて管理することを特徴とする分散ファイル共有システム。
【請求項３】　請求項１または２に記載のシステムにおいて、前記大域管理機能ブロックは、前記インデックス情報の参照および更新を制御することを特徴とする分散ファイル共有システム。
【請求項４】　請求項１、２または３に記載のシステムにおいて、該システムは、前記ホスト端末装置へのアクセス許可をユーザ情報として登録し、かつ該ユーザ情報の認証に基づいてアクセス許可の可否を管理するユーザ管理機能ブロックを前記ホスト端末装置に含むことを特徴とする分散ファイル共有システム。
【請求項５】　請求項１ないし４のいずれか一項に記載のシステムにおいて、前記ホスト端末装置は、前記大域管理機能ブロックの前記インデックス情報で管理される前記共有に関するファイルのキャッシュを保持し、かつ該ファイルの親ディレクトリが有する前記アクセス権限情報のキャッシュを保持することを特徴とする分散ファイル共有システム。
【請求項６】　請求項５に記載のシステムにおいて、前記各ホスト端末装置上の前記ファイル共有管理機能ブロックは、局所的に実施されたアクセス権限情報の更新を前記大域管理機能ブロックに通知し、
さらに、該大域管理機能ブロックは、該アクセス権限情報を格納する他のホスト端末装置すべてに対して該アクセス権限情報の更新を通知することを特徴とする分散ファイル共有システム。
【請求項７】　請求項１、２、５または６に記載のシステムにおいて、前記アクセス権限情報は、可変長であることを特徴とする分散ファイル共有システム。
【請求項８】　請求項７に記載のシステムにおいて、前記アクセス権限情報は、登録されているユーザ名またはグループ名に対する読出し権限、書込み権限、ならびに前記読出し権限および前記書込み権限のいずれかが含まれていることを特徴とする分散ファイル共有システム。
【請求項９】　請求項８に記載のシステムにおいて、前記読出し権限は、前記ディレクトリの直下にあるファイル名またはディレクトリ名の読出し、該ファイルに対する検索、該ファイルの前記キャッシュへの作成および該ファイルの読出しを許可することを特徴とする分散ファイル共有システム。
【請求項１０】　請求項８に記載のシステムにおいて、前記書込み権限は、前記ディレクトリの直下にファイルまたはディレクトリの新たな登録、該ファイルに対する更新または削除、前記ディレクトリの削除を許可することを特徴とする分散ファイル共有システム。
【請求項１１】　請求項８に記載のシステムにおいて、前記読出し権限および前記書込み権限は、前記ディレクトリの直下にあるファイル名またはディレクトリ名の読出し、該ファイルに対する検索、該ファイルの前記キャッシュへの作成および該ファイルの読出しの許可と、
前記ディレクトリの直下にファイルまたはディレクトリの新たな登録、該ファイルに対する更新または削除、前記ディレクトリの削除の許可とを含むことを特徴とする分散ファイル共有システム。
【請求項１２】　データをまとめて扱うファイルを階層構造の位置を示すディレクトリの下に格納してファイル管理するホスト端末装置のそれぞれがネットワーク接続されたホスト端末装置において該ディレクトリおよび前記ファイルのいずれか一方の情報に対する操作に応じて前記ファイル管理の対象にアクセス要求を出力する第１の工程と、
該アクセス要求に応じて前記対象へのアクセス操作するユーザの権限をアクセス権限とし、まとめて管理されているアクセス権限を用いて該ユーザの前記アクセス要求に対するアクセス資格があるか否かを判定する第２の工程と、
該アクセス資格を満足する前記アクセス権限が前記ユーザにあるとき、前記アクセス要求に対応した処理を前記情報に施す第３の工程と、
前記アクセス資格がないユーザと判定されたとき、前記アクセス要求に対応した処理を回避する応答を出力する第４の工程と、
前記ホスト端末装置のそれぞれに対する前記アクセス権限を含むインデックス情報を管理対象として該インデックス情報に対して処理を施す第５の工程と、
該インデックス情報の処理に対応した応答信号を前記操作したホスト端末装置に出力する第６の工程とを含むことを特徴とする分散ファイル共有システムのファイルアクセス制御方法。

【請求項１３】　請求項12に記載の方法において、前記アクセス権限は、登録されているユーザ名またはグループ名に対する読出し権限、書込み権限、ならびに前記読出し権限および前記書込み権限のいずれかが含まれていることを特徴とする分散ファイル共有システムのファイルアクセス制御方法。

【請求項１４】　請求項13に記載の方法において、前記読出し権限は、前記ディレクトリの直下にあるファイル名またはディレクトリ名の読出し、該ファイルに対する検索、該ファイルの前記キャッシュへの作成および該ファイルの読出しを許可することを特徴とする分散ファイル共有システムのファイルアクセス制御方法。

【請求項１５】　請求項13に記載の方法において、前記書込み権限は、前記ディレクトリの直下にファイルまたはディレクトリの新たな登録、該ファイルに対する更新または削除、前記ディレクトリの削除を許可することを特徴とする分散ファイル共有システムのファイルアクセス制御方法。

【請求項１６】　請求項13に記載の方法において、前記読出し権限および前記書込み権限は、前記ディレクトリの直下にあるファイル名またはディレクトリ名の読出し、該ファイルに対する検索、該ファイルの前記キャッシュへの作成および該ファイルの読出しの許可と、
前記ディレクトリの直下にファイルまたはディレクトリの新たな登録、該ファイルに対する更新または削除、前記ディレクトリの削除の許可とを含むことを特徴とする分散ファイル共有システムのファイルアクセス制御方法。

【請求項１７】　請求項12ないし16のいずれか一項に記載の方法において、第２の工程は、前記ホスト端末装置からユーザがアクセスした際に該アクセスしたユーザが前記ネットワーク全体を管理するシステム管理者か否かを判定する第７の工程と、
該判定が前記システム管理者の場合、前記ファイルに対する操作の可否をアクセス権限とし、該ファイルすべてのアクセス権限を許可する第８の工程と、
前記判定が前記システム管理者と異なるユーザの場合、該ユーザの属するグループに対するアクセス権限を取得する第９の工程と、
操作するホスト端末装置に操作対象の元を表す親ディレクトリ情報を保持しているか否かを判定する第10の工程と、
前記親ディレクトリ情報が空状態と判定した場合、前記ディレクトリを含めたファイルに対するインデックス情報の中から対応する親ディレクトリ情報を取得する第11の工程と、
前記親ディレクトリ情報が含むアクセス権限を確認し、該アクセス権限の確認結果に応じて前記アクセス権限の許可および禁止のいずれかを行う第12の工程とを含むことを特徴とする分散ファイル共有システムのファイルアクセス制御方法。

【請求項１８】　請求項17に記載の方法において、第11の工程と第12の工程の間に、前記ユーザが前記ディレクトリの所有者か否かを判定する第13の工程を含み、
前記ディレクトリの所有者のとき、第12の工程に進み、これ以外のとき、前記アクセス権限を禁止して終了することを特徴とする分散ファイル共有システムのファイルアクセス制御方法。

【請求項１９】　請求項12ないし18のいずれか一項に記載の方法において、第５の工程は、該処理期間中、前記アクセス要求を送出したホスト端末装置とだけ一対一の接続関係を保つ制御を行うことを特徴とする分散ファイル共有システムのファイルアクセス制御方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、分散ファイル共有システムおよびそのファイルアクセス制御方法に関し、分散ファイル共有システムは、ネットワークに分散して接続する複数の端末装置に備えるファイルの共有化を図って仮想的に一つのファイルシステムとする分散ファイルシステムに適用して好適なものであり、この分散ファイルシステムを仮想的なファイルシステムとして扱い、このファイルシステム内におけるディレクトリやファイルに対するアクセス制御に用いて好適なファイルアクセス制御方法に関するものである。

【０００２】

【従来の技術】通常、ホストとして用いる１台のコンピュータ端末装置（以下、ホスト端末装置という）上において、ファイルにアクセスする制御方法には、OS（OperatingSystem)機能としてのアクセス制御リストを使用する。OSの一つであるUNIX（登録商標）システムでは、ファイルやディレクトリ毎に、たとえば所有者、所有者グループ、その他のユーザに分類し、この分類に対して読出し、書込みおよび実行のアクセス権限をそれぞれに設定してファイルのアクセス制御を実現させている。ホスト端末装置を遠隔端末装置からネットワークを介して使用する場合、ホスト端末装置は、ホスト端末装置に格納されているユーザのアカウントに従って上述したファイルやディレクトリへのアクセス権限を判定している。

【０００３】この他に、アクセス制御の観点において、複数のホスト端末装置がファイルサーバを共有する場合、この共有にともなう負荷分散（負荷軽減）させる方法が提案されている。特開平11-120063号公報に記載の分散システムにおける共有ファイルシステムは、ファイル管理の計算負荷をなくし、ファイルデータの転送がネットワークを経由しないようにして処理の高速化を図っている。

【０００４】特開平9-305470号公報に記載の共有ファイル管理装置は、共有ファイルの状態およびファイル格納手段の状態を監視し、監視結果に基づきシステム管理部

で共有ファイルの配置または多重化の見直しを図り、ファイルアクセス制御部で見直しに応じた共有ファイルの移動または多重化を実行して運用効率を向上させている。また、特開平8-77054号公報に記載の分散ファイルシステムは、複数の計算機をネットワーク接続した計算機クラスタ上の複数のサーバ計算機に分散ファイルに対応する複数の分割ファイルの作成と削除を分割ファイル作成部と分割ファイル削除部とでそれぞれ行わせ、分散ファイル管理部では、分散ファイルに対する参照／更新要求を行う前に、分割ファイルに対する参照／更新振分け情報をクライアント計算機に配送し、この情報に応じて参照／更新要求振分け部では参照／更新要求の指定に用いるレコードのインデックスが格納される分割ファイルの所在を決定することで、処理要求が集中しても有効に負荷分散させている。

【０００５】さらに、特開平6-332782号公報に記載のファイルサーバシステムは、複数のファイルサーバがネットワークを介して並設され、ファイルサーバは各ファイル格納装置にアクセスする。このシステムにおいて、ファイルアクセス要求配分手段は、負荷情報モニタリング手段で計測した各ファイルサーバの負荷状況を参照し、ファイルアクセスするファイルサーバを選定するファイル管理を行い、選定したファイルサーバが自己のサーバまたは他のサーバか否かに応じて対応する通信制御手段を介してファイルアクセス要求を発行することにより、特に、ファイル書込み時に負荷の軽いファイルサーバを選んでアクセスさせて特定のファイルサーバにアクセス要求が集中しないようにしている。上述した４つの提案は、アクセス制御の観点に基づく提案であり、アクセス権限を付与してファイルのアクセスを制御する方法とまったく異なる技術である。

【０００６】

【発明が解決しようとする課題】ところで、複数のホスト端末装置がネットワークに分散して配設され、これらを仮想的に１つのファイルシステムのようにみなした分散ファイル共有システムについて検討する。一般的に、このような分散ファイルシステムでは、システム内におけるアクセス権限の一元管理をどのように行うかが大きな問題になる。すなわち、この分散ファイルシステムにおいて１台のホスト端末装置がアクセス権限管理サーバを動作制御する。このとき、前述したUNIXシステムのファイルアクセスに倣ってファイルパスのアクセス権限すべてをチェックする方法を適用すると、分散ファイルシステムは、ファイルのアクセス権限の判定にディレクトリおよびファイルの両方をチェックすることから、検索に時間がかかり過ぎて現実的でない場合がある。

【０００７】ファイルの共有性を加味した分散ファイル共有システムでファイル検索を行う場合、検索条件に合致するすべてのファイルパスの各ディレクトリおよびファイルに対してアプリケーションレベルでのアクセス権限の判定を行うことから、分散ファイル共有システムでは、さらに多くの検索時間がかかると推定される。

【０００８】また、分散ファイル共有システムは、上述したように仮想的に１つのファイルシステムとみなすことから、あるホスト端末装置のファイルの複製を他のホスト端末装置にキャッシュすることが可能になる。このキャッシュ機能により、一度アクセスして得られたホスト端末装置のファイルは、ファイルが削除されない限りこのファイルに関するアクセス権限の確認を行うためネットワークを介して行わなくてもよくなる。この状況にもかかわらず、ホスト端末装置は、内蔵するファイルのアクセス権限を判定する際にネットワークを介してアクセス権限管理用のサーバに問い合わせを行ってしまう。これは、ホスト端末装置にファイルをキャッシュする意味を減殺させてしまう。

【０００９】本発明はこのような従来技術の欠点を解消し、アクセス権限に関わる検索時間の短縮化およびファイル操作の効率化を図ることができる分散ファイル共有システムおよびそのファイルアクセス制御方法を提供することを目的とする。

【００１０】

【課題を解決するための手段】本発明は上述の課題を解決するために、データをまとめて扱うファイルを階層構造の位置を示すディレクトリの下に格納してファイル管理する端末装置を備え、この端末装置のそれぞれをネットワークに接続させ、各端末装置をホスト端末装置とし、ホスト端末装置は、これらのホスト端末装置のうち、共有に関するディレクトリおよびファイルにアクセスする資格の情報をアクセス権限情報としてインデックス情報に含め、このインデックス情報すべてを管理する大域管理機能ブロックをネットワーク内に少なくとも一つ含み、ホスト端末装置は、各ホスト端末装置で共有するディレクトリを含めたファイルの管理およびこの各ホスト端末装置自体に対する局所的なファイル管理、ならびにファイル管理に用いるディレクトリに対するアクセス権限情報の設定および管理を行うファイル共有管理機能ブロックを含むことを特徴とする。

【００１１】本発明の分散ファイル共有システムは、大域管理機能ブロックに共有するファイルに関するインデックス情報を用いてインデックス情報からアクセス権限に応じた管理を行い、ユーザが操作するホスト端末装置が対象のディレクトリ情報を持っていなくても、各ホスト端末装置へのアクセスを行わずに大域管理機能ブロックから容易にアクセス権限を得ることができ、ファイル共有管理機能ブロックで局所的な管理を行う。

【００１２】また、本発明は上述の課題を解決するために、データをまとめて扱うファイルを階層構造の位置を示すディレクトリの下に格納してファイル管理するホスト端末装置のそれぞれがネットワーク接続されたホスト端末装置においてこのディレクトリおよびファイルのい

ずれか一方の情報に対する操作に応じてファイル管理の対象にアクセス要求を出力する第１の工程と、このアクセス要求に応じて対象へのアクセス操作するユーザの権限をアクセス権限とし、まとめて管理されているアクセス権限を用いてこのユーザのアクセス要求に対するアクセス資格があるか否かを判定する第２の工程と、このアクセス資格を満足するアクセス権限がユーザにあるとき、アクセス要求に対応した処理を情報に施す第３の工程と、アクセス資格がないユーザと判定されたとき、アクセス要求に対応した処理を回避する応答を出力する第４の工程と、ホスト端末装置のそれぞれに対するアクセス権限を含むインデックス情報を管理対象としてこのインデックス情報に対して処理を施す第５の工程と、このインデックス情報の処理に対応した応答信号を操作したホスト端末装置に出力する第６の工程とを含むことを特徴とする。

【００１３】本発明の分散ファイル共有システムのファイルアクセス制御方法は、ディレクトリおよびファイルのいずれか一方の情報に対する操作を行う際に、この操作に対する対象へのアクセスができるか否かを設定されているアクセス権限により判定して制御することにより、個々のアプリケーションレベルでのファイル検索による判定に比べて検索の所要時間の短縮化を図ることができる。

【００１４】

【発明の実施の形態】次に添付図面を参照して本発明による分散ファイル共有システムの一実施例を詳細に説明する。

【００１５】本実施例は、本発明の分散ファイル共有システムを仮想ファイルシステムである分散ファイルシステム10に適用した場合である。本発明と直接関係のない部分について図示および説明を省略する。以下の説明で、信号はその現れる接続線の参照番号で指示する。

【００１６】分散ファイルシステム10には、図１に示すように、複数のホスト端末装置12, 14, 16, 18, 20, ･･･がネットワーク100に接続されている。ホスト端末装置12には、ファイル共有インデックス管理機能部120およびインデックス情報ファイル1244が含まれている。ホスト端末装置14には、ユーザ情報管理機能部140およびユーザ情報ファイル1444が備えられている。また、ホスト端末装置16, 18には、それぞれ、ファイル共有管理機能部160, 180およびデータファイル1644, 1844が一組ずつ配設されている。参照符号が付されていない一点鎖線で表したホスト端末装置も３種類のいずれかが該当する。

【００１７】さらに、各ホスト端末装置について構成を説明する。ホスト端末装置12, 14, 16, 18, ･･･には、本体装置部12a, 14a, 16a, 18a, ･･･および周辺装置部12b, 14b, 16b, 18b, ･･･が、それぞれ備えられている。ホスト端末装置12において本体装置部12aは、図２に示すように、CPU (Central Processing Unit)120a、メモリ122、ストレージ124およびネットワーク接続部126を含む。周辺装置部12bには、ディスプレイ128、キーボード130、およびポインティングデバイスとしてマウス132があり、周辺装置部12bは本体装置部12aに信号線134、136、138を介して接続されている。

【００１８】ストレージ124は、ハードディスクドライブ装置（Hard Disk Drive unit：以下、HDDという）である。ストレージ124は、プログラム記憶領域124Pとデータ記憶領域124Dとに分けて記憶している。プログラム記憶領域124Pには、ファイル共有インデックス管理プログラム1240および操作インタフェース処理プログラム1242が格納されている。ファイル共有インデックス管理プログラム1240は、ネットワーク100に接続するホスト端末装置12, 14, 16, 18, ･･･において共有指定されたファイルやディレクトリすべてのインデックス情報ファイル1244を管理する。データ記憶領域124Dには、インデックス情報ファイル1244があり、共有指定されたファイルやディレクトリすべてのインデックス情報が含まれている。

【００１９】ネットワーク接続部126は、ネットワーク100との接続インタフェースである。ホスト端末装置12は、ネットワーク接続部126を介してネットワーク100に接続され、分散ファイルシステム10は、他のホスト端末装置14, 16, 18, ･･･を含めた、全体として一つの仮想ファイルシステムを構築している。

【００２０】ホスト端末装置14において本体装置部14aは、図３に示すように、CPU 140a、メモリ142、ストレージ144およびネットワーク接続部146を含む。周辺装置部14bには、ディスプレイ148、キーボード150、およびマウス152があり、周辺装置部14bは本体装置部14aと信号線154、156、158を介して接続されている。

【００２１】ストレージ144は、HDDで、ストレージ144は、プログラム記憶領域144Pとデータ記憶領域144Dとに分けて記憶している。プログラム記憶領域144Pには、ユーザ情報管理プログラム1440および操作インタフェース処理プログラム1442が格納されている。ユーザ情報管理プログラム1440は、ネットワーク100に接続するホスト端末装置12, 14, 16, 18, ･･･におけるユーザ情報ファイル1444を管理する。ユーザ情報ファイル1444には、たとえば、ユーザを特定するユーザID（IDentifier）、パスワード等の秘密情報がある。データ記憶領域144Dには、ユーザ情報ファイル1444が格納されている。

【００２２】ユーザ情報管理機能部140は、ユーザ情報ファイル1444内のユーザ情報を定義する情報を基にグループ化することができる。定義情報は、様々に設定できることから、ユーザは複数のグループに所属することができる。ユーザ情報管理機能部140は、操作するユーザが操作インタフェースを使用する場合、ユーザ認証を行う。ユーザ認証は、上述したユーザIDおよび秘密情報がログイン時にキーボードを用いてホスト端末装置に入力

され、ユーザ情報ファイル142内のユーザ情報と照合することで行われる。
【0023】より具体的に、ユーザ情報とは、ユーザIDおよび秘密情報等である。ユーザ情報管理機能部140は、この認証の結果、一致した正当なユーザにだけ操作を許可する。このとき、該当するホスト端末装置の操作インタフェース処理プログラムがメモリ122にロードされ、操作インタフェース処理機能部142aとして操作を受け付ける。
【0024】ネットワーク接続部146は、ネットワーク100との接続インタフェースである。ユーザ情報管理機能140およびユーザ情報ファイル142を持たないホスト端末装置12, 16, 18, ･･･では、ネットワーク接続部146を介してそれぞれのホスト端末装置12, 16, 18, ･･･からのユーザIDおよび秘密情報が供給され、認証結果の可否が返送される。この返送された結果に応じてユーザのアクセス可否が決まる。
【0025】なお、ホスト端末装置14には、図示しないが後述するファイル共有管理機能部を含んでいてもよい。
【0026】また、ホスト端末装置16において本体装置部16aは、図4に示すように、CPU 160a、メモリ162、ストレージ164およびネットワーク接続部166を含む。周辺装置部16bには、ディスプレイ168、キーボード170、およびマウス172があり、周辺装置部16bは本体装置部16aと信号線174、176、178を介して接続されている。
【0027】ストレージ164は、HDDでストレージ164は、プログラム記憶領域164Pとデータ記憶領域164Dとに分けて記憶している。プログラム記憶領域164Pには、ファイル共有管理プログラム1640および操作インタフェース処理プログラム1642が格納されている。ファイル共有管理プログラム1640は、ネットワーク100に接続するホスト端末装置16, 18, ･･･において個々に共有指定されたデータファイル1644, 1844に対する処理に応じて管理する。 データ記憶領域164Dのデータファイル1644には、たとえばディレクトリ情報やファイル情報が格納されている。
【0028】ファイル共有管理機能部160は、ファイルやディレクトリの操作機能を局所的に実現させるとともに、他のホスト端末装置12, 14, 18, ･･･のファイル共有インデックス管理機能部120、ユーザ情報管理機能部140およびファイル共有管理機能部180, ･･･との連携を図ることができる。ファイルやディレクトリの操作機能とは、後段で説明するように登録機能、検索機能、獲得機能、読出し機能、更新機能および削除機能である。
【0029】分散ファイルシステム10には、ホスト端末装置16と同様の構成のホスト端末装置18, ･･･が分散して配置されている。ホスト端末装置18は、ファイル共有管理機能部180を有している。
【0030】次に分散ファイルシステム10で扱われるデータ記憶領域164D, 184Dおよび124Dの情報について説明する。ホスト端末装置16は、データ記憶領域164Dにデータファイル1644を含む。データファイル1644は、図5(a)に示すように、ディレクトリ“/P”に対するディレクトリ情報Pおよびファイル164A（＝“/P/A”）を含む。ここで、ファイル164Aは、ファイル情報Aとファイル164aを一つのファイルとして保持している。ディレクトリ“/P”およびファイル164Aは、操作インタフェース処理機能部162aを通じて操作され、ファイル共有管理機能部160の機能により作成・登録されたものである。
【0031】ただし、ファイル164aは、ホスト端末装置16で用いるOSにおけるファイルパス“/P/A”のファイルというわけではない。操作インタフェース処理機能部162aからファイル共有管理機能部160の管理下にあるファイルを表示させると、図5(b)のローカルビューのように、ディレクトリ“/P”に対してファイル164Aのツリー構造を示す。
【0032】同様に、ファイル共有管理機能部180を含むホスト端末装置18は、図6(a)のデータ記憶領域にディレクトリ情報Q, Rがあり、ファイル184Bがファイル情報Bおよびファイル184b、ファイル184Cがファイル情報Cおよびファイル184cで、それぞれ関連付けている。より具体的には、ディレクトリ“/Q”，“/Q/R”およびファイル“/Q/184B”，“Q/R/184C”の関係にファイルを管理している。上述したディレクトリおよびファイルは、それぞれ操作インタフェース処理機能部182aを通じて操作され、ファイル共有管理機能部180の機能により作成・登録されたものである。この場合もファイル184b, 184cは、ホスト端末装置18で用いるOSにおけるファイルパス“/Q/184B”，“/Q/R/184C”のファイルというわけではない。操作インタフェース処理機能部182aからファイル共有管理機能部180の管理下にあるファイルを表示させると、図6(b)のローカルビューのように、ディレクトリ“/Q”に対してファイル184Bとディレクトリ“/Q/R”に対してファイル184Cのツリー構造を示す。
【0033】また、ホスト端末装置12は、図7(a)に示すように分散ファイルシステム10のディレクトリ情報P, Q, Rおよびファイル情報A, B, Cすべてをファイル共有インデックス管理機能部120で一元管理する。これらすべてに対してインデックス情報と呼び、インデックス情報ファイル1244に格納されている。ファイル共有インデックス管理機能部120の管理によってインデックス情報ファイル1244をグローバルビュー表示させると、ディレクトリ情報とファイル情報とを含むインデックス情報ファイル1244が、図7(b)に示すように、ディレクトリ“/P”-ファイル情報A, ディレクトリ“/Q”-ファイル情報B, およびディレクトリ“/Q/R”-ファイル情報Cの仮想的なツリー構造で表される。
【0034】次に前述したディレクトリ情報およびファイル情報について説明する。ディレクトリ情報は、図8

に示すようにキー40、ディレクトリ名42、ディレクトリ更新日44、所有者46、破棄日48、公開日50、アクセス権限52を有している。キー40は、たとえばディレクトリを表す値である。ディレクトリ名42は、分散ファイルシステム10における仮想ディレクトリパス名を示す。ディレクトリ更新日44は、このディレクトリが最後に更新された日の情報である。所有者46は、ディレクトリの作成者を所有者として所有者名を表す。ディレクトリ情報は、この所有者とシステム管理者adminだけが更新することができる。

【００３５】また、検索対象外を指定する日として破棄日48と公開日50がある。破棄日48は、この日を過ぎた場合、検索対象外にする検索対象の有効期限終了日で、公開日50は、これ以前の日を検索対象外とする検索対象の有効期限開始日である。アクセス権限52は、アクセス制御リストを含む。

【００３６】アクセス制御リストは、図９に示すように、ユーザ名またはグループ名とアクセス権限とを対応させたものである。ユーザ名およびグループ名は、ユーザ情報管理部機能部140によってユーザ情報ファイル1444に登録されている名称である。アクセス権限には、読出し（以下、rという）権限、書込み（以下、wという）権限および読出し書込み権限（以下、rwという）権限があり、これら３つの内、いずれか一つの権限がユーザ名またはグループ名に対応させる。アクセス制御リストは、可変長にしている。アクセス権限については後段でさらに説明する。

【００３７】ファイル情報は、図10に示すように、キー60、ファイル名62、ファイル更新日64、所有者66、破棄日68、公開日70、ファイルの状態72を含んでいる。ファイル情報においてディレクトリ情報と同じ名称の項目は、同じ定義であることから説明を省略する。ここで、ファイル名62は仮想ファイルパス名、所有者66はファイルの登録者である。また、ファイルの状態72とは、そのファイルがオリジナルかキャッシュ、すなわち複製されたものかを区別する識別子である。

【００３８】ここで、ホスト端末装置16がホスト端末装置18からファイル184Bをキャッシュすることにより獲得（複製）した場合のファイル共有管理を図11に示す。この獲得にともなってホスト端末装置16のデータ記憶領域164Dには、図11(a)に示すようにディレクトリ情報Qとファイル184Bが追加される。ファイル184Bは、ファイル情報Bおよびファイル184bを含んでいる。キャッシュされた情報には、斜線が施されている。したがって、ユーザが操作インタフェース機能部162aからファイル共有管理機能部160の管理下のファイル表示を行わせると、ローカルビューでディレクトリ“/P”にファイル164Aとディレクトリ“/Q”にファイル184Bが見える。

【００３９】ただし,ファイル184Bは、斜線で示されるようにキャッシュ（複製）であることも示している。

【００４０】次にディレクトリ情報におけるアクセス権限について説明する。たとえば、ディレクトリPにおけるr権限とは、任意のホスト端末装置のファイル共有管理機能部からディレクトリPに対する権限で、第１に、ディレクトリP直下のファイル名の読出し権限、第２に、ディレクトリP直下のファイルに対する検索/獲得/読出し処理の権限、そして、第３に、ディレクトリP直下のディレクトリ名の読出し権限である。上述した獲得処理は、読み出したファイルのキャッシュを作成することを意味し、読出し処理は、単なる読出しを意味するものである。

【００４１】また、ディレクトリPにおけるw権限とは、任意のホスト端末装置のファイル共有管理機能部からディレクトリPに対する権限で、第１に、ディレクトリP直下に新たなディレクトリの作成権限、第２に、ディレクトリP直下のディレクトリの削除権限、第３に、ディレクトリP直下に新たなファイルの登録権限、そして、第４に、ディレクトリP直下のファイルの更新／削除権限である。ただし、キャッシュファイルの更新はできない。

【００４２】分散ファイルシステム10では、システム管理者アカウントadminが存在し、アカウントadminは、すべてのディレクトリに対してrw権限を有している。また、ディレクトリの所有者は、所有するディレクトリに対してrw権限を持っている。

【００４３】次にアクセス権限の判定手順について説明する。ユーザは、ディレクトリ操作やファイル操作を行う前にアクセス権限の判定を受けてアクセス操作の可否を得ている。図12に示すように、操作ユーザが、システム管理者か否かの判定を行う（ステップS10）。実際に、この判定は、ユーザIDがシステム管理者アカウントadminか否かで判定する。ユーザIDがシステム管理者アカウントadminに一致した場合（YES）、操作ユーザに対してディレクトリおよびファイルすべてのアクセスが許可される（ステップS12）。このときのアクセス権限は、rw権限である。この判定の後、終了に移行する。

【００４４】また、操作ユーザがシステム管理者アカウントadminでなかった場合（NO）、分散ファイルシステム10において操作ユーザの属するグループすべての取得を行う（ステップS14）。このグループ情報は、使用するホスト端末装置にログインした際にホスト端末装置14のユーザ情報管理機能部140を動作させて、ユーザ情報ファイル1444から取得することができる。

【００４５】次に操作しているホスト端末装置上に操作対象とするディレクトリやファイルの親ディレクトリ情報が有るか否かの判定を行う（ステップS16）。親ディレクトリ情報を保持していない場合（NO）、操作中のホスト端末装置は、ホスト端末装置12のファイル共有インデックス管理機能部120を介してインデックス情報ファイル1244から親ディレクトリ情報を取得する（ステップS1

8)。

【0046】また、親ディレクトリ情報を保持している場合や親ディレクトリ情報を取得した後に、親ディレクトリ情報に含まれるアクセス制御リストを参照して、アクセス権限の確認を行う（ステップS20)。

【0047】次に確認で得られたアクセス権限に一つでも許可を示すエントリが存在するか否かを判定する（ステップS22)。許可エントリが存在する場合（YES)、操作中のホスト端末装置にて許可に応じた処理が行えるようにする（許可：ステップS24)。また、許可エントリが存在しない場合（NO)、ディレクトリおよびファイルへのアクセスを禁止する（ステップS26)。これらの処理を経て終了に移行する。

【0048】アクセス権限の判定には、上述した手順だけでなく、図13ディレクトリ情報を更新する場合に行う判定処理もある。図12のアクセス権限の確認手順と同じ処理を行う過程には、図12のフローチャートで付した参照符号を付し、説明を大幅に省略する。システム管理者か否かの判定（ステップS10)によって、操作ユーザがシステム管理者の場合、すべて許可してユーザ認証を終了する。これ以外の操作ユーザがシステム管理者でないと判定された場合（NO）、操作対象の親ディレクトリ情報を保持しているか否かの判定を行う（ステップS16)。この親ディレクトリを保持していない場合（NO）、ファイル共有インデックス管理機能部120を動作させてインデックス情報ファイル1244から取得する。

【0049】次に保持しているディレクトリ情報または取得したディレクトリ情報を参照して、操作ユーザがディレクトリの所有者か否かの確認を行う（ステップS28)。ディレクトリ情報の所有者と操作ユーザのユーザIDとを比較して一致した場合（YES)、情報の更新を許可する（ステップS24)。また、この比較が不一致の場合（NO)、情報の更新を禁止し、不許可にする（ステップS26)。このようにして情報更新の場合、ユーザ認証を行って更新資格の有無を判定して終了する。

【0050】このようなアクセス権限の判定およびユーザ認証を行いながら、分散ファイルシステム10におけるディレクトリ操作およびファイル操作のシーケンスを説明する。ディレクトリおよびファイルに関するシーケンスは、基本的に図14に示す通りである。この基本的なシーケンスは、ディレクトリの作成、ディレクトリ情報の更新およびディレクトリの削除である。

【0051】図14のシーケンスでは、説明の便宜上、ホスト端末装置16を用いている。ディレクトリに対する操作指示信号200が、時刻T10にてキー操作により操作インタフェース処理機能部162aに供給される。操作インタフェース処理機能部162aは、時刻T12に指示要求信号202をファイル共有管理機能部160に送出する。

【0052】ファイル共有管理機能部160では、時刻T14でディレクトリに対するアクセス権限の判定処理を開始する。ファイル共有管理機能部160は、時刻T16でディレクトリ情報がない場合、ホスト端末装置12のファイル共有インデックス管理機能部120に親ディレクトリ情報の情報取得要求信号204を送出する。ファイル共有インデックス管理機能部120は、時刻T18で一元管理しているインデックス情報ファイル1244を基に要求された親ディレクトリ情報の検索を開始する。ファイル共有インデックス管理機能部120は、検索によって得られた親ディレクトリ情報を時刻T20に取得情報信号206としてファイル共有管理機能部160に送出する。

【0053】ファイル共有管理機能部160は、時刻T22で供給された親ディレクトリ情報を基にアクセス権限の確認を行う。ファイル共有管理機能部160は、確認により得られたアクセス権限がないと判定した場合、時刻T24で権限がないことを示す応答信号208を操作インタフェース処理機能部162aに送出する。さらに、操作インタフェース処理機能部162aは、ユーザが使用するディスプレイ168に権限のないことを示す応答信号210を出力する。これにより、ユーザはアクセス権限のないことを知る。

【0054】また、アクセス権限の判定処理においてファイル共有管理機能部160では管理下のデータファイル1644に親ディレクトリ情報が存在した場合、時刻T16でただちにアクセス権限の確認を行う。アクセス権限のないことが判定されたとき、以降の処理を時刻T24にて行う応答処理に進み、さらに、上述した時刻T24，T26の応答処理を行う。

【0055】一方、アクセス権限の確認判定の結果、アクセス権限がある場合、時刻T16または時刻T22の段階から時刻T28での要求処理に進む。このとき、ファイル共有管理機能部160は、指示要求に応じたそれぞれの処理を行い、時刻T28で処理されたディレクトリ情報を情報通知信号212としてファイル共有インデックス管理機能部120に通知する。分散ファイルシステム10は、ネットワーク100を介して複数のホスト端末装置から同時に操作要求がファイル共有インデックス管理機能部120に出されても不整合が発生してしまう虞がある。

【0056】この不整合の発生を防止するため、ホスト端末装置120は、一つのホスト端末装置だけと通信を行う一対一の接続制御を行う。別な観点で見ると、ファイル共有インデックス管理機能部120は、接続中のホスト端末装置以外と通信できないように排他制御が行われている。この排他制御は、時刻T30から管理対象に対する処理が完了し、ファイル共有インデックス管理機能部120が応答信号214をファイル共有管理機能部160に送出する時刻T32まで継続する。

【0057】ファイル共有管理機能部160は、時刻T34に要求処理が完了したことを示す応答信号216を操作インタフェース処理機能部162aに出力する。さらに、操作インタフェース処理機能部162aは、時刻T36に応答信号218をディスプレイ168に送出する。これにより、操作ユー

ザは、所望の処理が完了したことを知る。
【００５８】より具体的な例を挙げて簡単に説明する。分散ファイルシステム10においてホスト端末装置16からディレクトリ作成の場合、ホスト端末装置16は、操作インタフェース処理機能部162aを介してディレクトリ作成の操作を行う（時刻T12）。ファイル共有管理機能部160は、アクセス権限のうち、w権限が操作ユーザにあるか否かの判定処理を開始する（時刻T14）。
【００５９】w権限が確認された場合、時刻T28までにファイル共有管理機能部160は、キーボード170から操作インタフェース処理機能部162aを介して供給された入力データを基にディレクトリ情報を作成する。ファイル共有管理機能部160は、時刻T28に作成したディレクトリ情報をファイル共有インデックス管理機能部120に通知する。時刻T30～T32の間にファイル共有インデックス管理機能部120は、前述した排他制御を行いながら、通知されたディレクトリ情報を管理対象としてインデックス情報ファイル1244に追加し、インデックス情報を更新する。
【００６０】時刻T16または時刻T22にてw権限がないと確認された場合、ファイル共有管理機能部160は、時刻T24の処理に進み、以後の応答処理を行う。
【００６１】ディレクトリ情報の更新を行う場合、ホスト端末装置16は、操作インタフェース処理機能部162aを介してディレクトリ情報更新操作を行う。この操作を受けてファイル共有管理機能部160は、ディレクトリ情報を所有しているか否かに応じて時刻T16および時刻T22のいずれかでユーザ認証のアクセス判定処理を行う。操作ユーザがディレクトリの所有者またはシステム管理者という判定された場合、ホスト端末装置16は、ディレクトリ情報の更新を行うアクセスが許可される。
【００６２】ファイル共有管理機能部160は、時刻T28までに、キーボード170から操作インタフェース処理機能部162aを介して供給された入力データを基に更新するディレクトリ情報を作成する。ファイル共有管理機能部160は、時刻T28に作成したディレクトリ情報をファイル共有インデックス管理機能部120に通知する。時刻T30～T32の間にファイル共有インデックス管理機能部120は、前述した排他制御を行いながら、通知されたディレクトリ情報を管理対象としてインデックス情報ファイル1244に追加し、インデックス情報を更新する。
【００６３】ところで、ファイル共有インデックス管理機能部120は、どのホスト端末装置に更新するディレクトリが存在するかを知っているので、更新されたディレクトリ情報を通知する。この通知をイベント通知という。イベント通知は、更新に限定されるものでなく、後述するディレクトリ削除、ファイル更新およびファイル削除においても行われる。以下に示す手順は、ディレクトリ削除、ファイル更新およびファイル削除においても同じであることから、上述の３つの処理に関して後段における説明は簡略化し、異なる点に着目して説明を行う。
【００６４】このイベント通知に該当するディレクトリをホスト端末装置18，20が持っているとする。この場合、図15に示すように、ファイル共有インデックス管理機能部120は、ホスト端末装置18，20にそれぞれ時刻T40，T44で更新されたディレクトリ情報をイベント通知信号220，222として通知する。ファイル共有管理機能部180は、時刻T42にイベント通知を受けて、時刻T48までにデータファイル1844内のディレクトリ情報の更新処理を行う。また、ファイル共有管理機能部20aは、時刻T46にイベント通知を受けて、時刻T50までにデータファイル（図示せず）ディレクトリ情報の更新処理を行う。
【００６５】ファイル共有管理機能部180は、時刻T48にファイル共有インデックス管理機能部120に更新したディレクトリ情報を更新通知信号224として通知する。同様に、ファイル共有管理機能部20aは、時刻T50に更新したディレクトリ情報を更新通知信号226として通知する。ファイル共有インデックス管理機能部120は、たとえば、時刻T52～T54の間を排他制御し、供給された更新通知信号224，226を基にホスト端末装置18，20のインデックス情報ファイル1244のインデックス情報を更新する。ファイル共有インデックス管理機能部120は、ホスト端末装置18，20にそれぞれ時刻T54，T56にて応答信号228，230を出力する。このように時刻T40～T56までの処理によりイベント通知に対応したディレクトリ情報の更新が行われる。
【００６６】ここで、上述した一連のイベント通知、更新処理、インデックス情報の更新、応答は、ホスト端末装置18，20に対して順序を考慮して行っているが、各ホスト端末装置に対する一連の手順が遵守されていれば、ホスト端末装置の順序は順不同にしてもよい。
【００６７】ディレクトリ情報を削除する場合、ホスト端末装置16は、操作インタフェース処理機能部162aを介してディレクトリの削除操作を行う（時刻T12）。ファイル共有管理機能部160は、アクセス権限のうち、w権限が操作ユーザにあるか否かの判定処理を開始する（時刻T14）。w権限が確認された場合、時刻T28までにファイル共有管理機能部160は、キーボード170から操作インタフェース処理機能部162aを介して供給された入力データまたはマウス172を介してディスプレイ168上で指摘された位置のディレクトリに対するディレクトリ情報を削除する。
【００６８】ファイル共有管理機能部160は、時刻T28に削除したディレクトリ情報をファイル共有インデックス管理機能部120に通知する。時刻T30～T32の間にファイル共有インデックス管理機能部120は、前述した排他制御を行いながら、通知されたディレクトリ情報を管理対象としてインデックス情報ファイル1244から削除する。ファイル共有インデックス管理機能部120は、ディレク

トリ削除のイベント通知も行う。
【００６９】ファイルに関するシーケンスも基本的に図14に示したシーケンスで行われる。ファイル共有管理機能部160は、ファイルの登録、検索・獲得（読出し）、更新、削除を処理として行う。ファイルを登録する場合、操作インタフェース処理機能部162aを介してファイルの登録操作を行う。ファイル共有管理機能部160では、アクセス権限の内、w権限に対する操作ユーザの判定処理を行う。アクセス権限がない場合、ファイル共有管理機能部160は、時刻T24で権限のないことを示す応答信号208を操作インタフェース処理機能部162aに出力する。また、w権限があると判定した場合、ファイル共有管理機能部160は、データファイル1644にファイルを登録する。この際、ファイル共有管理機能部160は、ファイル情報を作成し、登録するファイルを管理対象とする。ファイル共有管理機能部160は、時刻T28にて作成したファイル情報を情報通知信号212としてファイル共有インデックス管理機能部120に通知する。
【００７０】ファイル共有インデックス管理機能部120は、排他制御を行い、供給されたファイル情報を管理対象としてインデックス情報ファイル1244に追加してインデックス情報を更新する。この更新後、時刻T32にてファイル共有インデックス管理機能部120は、ファイル共有管理機能部160に応答信号214を出力する。
【００７１】ファイルを更新する場合、操作インタフェース処理機能部162aを介してファイルの更新操作を行う。ファイル共有管理機能部160では、アクセス権限の内、w権限に対する操作ユーザの判定処理を行う。アクセス権限がない場合、ファイル共有管理機能部160は、時刻T24で権限のないことを示す応答信号208を操作インタフェース処理機能部162aに出力する。
【００７２】また、w権限があると判定した場合、ファイル共有管理機能部160は、データファイル1644に対してファイルの更新を行う。この際、ファイル共有管理機能部160は、更新に応じたファイル情報を作成し、登録するファイルを管理対象とする。ここで、ファイル共有管理機能部160が更新可能なファイルは、オリジナルファイルだけである。ファイル共有管理機能部160で複製により作成されたキャッシュファイルは、更新できない。キャッシュファイルの更新は、後述するようにファイル共有インデックス管理機能部120からのイベント通知を受けて対象のホスト端末装置に獲得要求を出力して行われる。
【００７３】ファイル共有管理機能部160は、時刻T28にて作成したファイル情報を情報通知信号212としてファイル共有インデックス管理機能部120に通知する。ファイル共有インデックス管理機能部120は、排他制御を行い、供給されたファイル情報を管理対象としてインデックス情報ファイル1244のインデックス情報を更新する。この更新後、時刻T32にてファイル共有インデックス管理機能部120は、ファイル共有管理機能部160に応答信号214を出力する。
【００７４】ところで、ファイル共有インデックス管理機能部120は、更新したキャッシュファイルがどのホスト端末装置に含まれているか一元管理している。これにより、ファイル共有インデックス管理機能部120は、前述したシーケンスにより更新対象のホスト端末装置18, 20に対してイベント通知する。
【００７５】なお、このイベント通知処理の場合、図15のシーケンスに含まれていないシーケンスが行われる。このシーケンスでは、ファイル共有管理機能部180, 20aが、ファイル共有管理機能部160に対して更新ファイルの獲得要求を出力する。ファイル共有管理機能部160は、供給される獲得要求に応じて更新したファイル情報および更新ファイルをファイル共有管理機能部180, 20aに出力する。ファイル共有管理機能部180, 20aは、供給される更新したファイル情報および更新ファイルに応じてキャッシュファイルを更新する。ファイル共有管理機能部180, 20aは、それぞれ、ファイル共有インデックス管理機能部120に更新通知信号224, 226として更新したファイル情報を通知する。ファイル共有インデックス管理機能部120は、排他制御を行い、ファイル共有管理機能部180, 20aから供給されたファイル情報によりインデックス情報ファイル1244のインデックス情報を更新する。この更新後、ファイル共有インデックス管理機能部120は、時刻T54, T56にてそれぞれ、応答信号228, 230を出力する。
【００７６】次にファイルの削除を行う場合、操作インタフェース処理機能部162aを介してファイルの削除操作を行う。ファイル共有管理機能部160では、アクセス権限の内、w権限に対する操作ユーザの判定処理を行う。アクセス権限がない場合、ファイル共有管理機能部160は、時刻T24で権限のないことを示す応答信号208を操作インタフェース処理機能部162aに出力する。また、w権限があると判定した場合、ファイル共有管理機能部160は、データファイル1644に対してファイルの削除を行う。ファイル管理機能部160は、削除したファイル情報を情報通知信号212としてファイル共有インデックス管理機能部120に通知する。ホスト端末装置16におけるキャッシュファイルの削除は、この通知段階までである。
【００７７】ホスト端末装置16におけるオリジナルファイルを削除する場合、さらに、ファイル共有インデックス管理機能部120は排他制御を行い、ファイル共有管理機能部160から供給されたファイル情報をインデックス情報ファイル1244から削除する。ファイル共有インデックス管理機能部120は、削除したファイルに関して複製されたキャッシュファイルの存在するホスト端末装置を一元管理して知っているから、前述した更新と同様にイベント通知し、ファイル共有管理機能部180, 20aにおける削除したファイルのファイル情報およびファイルを削

除する。ファイル共有インデックス管理機能部120は、ファイル共有管理機能部180，20aからの削除したファイル情報の通知を受けてインデックス情報ファイル1244からホスト端末装置18，20のファイル情報を削除する。

【００７８】最後に、ファイルの検索、獲得（読出し）を行う場合について図16を参照しながら、説明する。ホスト端末装置16は、操作インタフェース処理機能部162aにファイル検索を指示する操作指示信号200を入力する（時刻T10）。操作インタフェース処理機能部162aは、ファイル検索要求として指示要求信号202をファイル共有管理機能部160に出力する（時刻T12）。ファイル共有管理機能部160は、アクセス権限のうち、r権限が操作ユーザにあるか否かの判定処理を開始する（時刻T14）。ここで、ファイル共有管理機能部160は、操作ユーザに対するディレクトリ情報の有無を調べる。

【００７９】ファイル共有管理機能部160で操作ユーザのディレクトリ情報がない場合、ファイル共有管理機能部160は、ファイル共有インデックス管理機能部120に情報取得要求信号204を出力する（時刻T16）。ファイル共有インデックス情報管理機能部120では、時刻T18以降インデックス情報ファイル1244から操作ユーザのアクセス権限リストの検索を行う。ファイル共有インデックス管理機能部120は、インデックス情報ファイル1244から時刻T18～T20の間にアクセス権限リストを含んだ該当するディレクトリ情報を取得する。取得したディレクトリ情報は、ファイル共有インデックス管理機能部120からファイル共有管理機能部160に取得情報信号206として供給される（時刻T22）。

【００８０】また、ファイル共有管理機能部160が操作ユーザのディレクトリ情報を持っている場合、ファイル共有管理機能部160は、時刻T16にてアクセス権限がr権限か否かの判定処理を行う。したがって、ファイル共有管理機能部160は、時刻T16および時刻T22のいずれかにて供給されるディレクトリ情報からアクセス権限の内、r権限があるか否かの判定処理を行う。r権限がないと判定した場合、ファイル共有管理機能部160は、時刻T24で権限がないことを示す応答信号208を操作インタフェース処理機能部162aに出力する。操作インタフェース処理機能部162aは、応答信号208を受けて時刻T26に応答信号210をディスプレイ168に出力する。

【００８１】r権限が確認された場合、時刻T28までにファイル共有管理機能部160は、キーボード170から操作インタフェース処理機能部162aを介して供給された検索データを情報通知信号212としてファイル共有インデックス管理機能部120に通知する。時刻T30～T32の間にファイル共有インデックス管理機能部120は、前述した排他制御を行いながら、通知された検索データに対するインデックス情報ファイル1244にて管理しているファイル情報の検索を行う。ファイル共有インデックス管理機能部120は、検索データに一致するファイル情報の照合により、所望するファイルがホスト端末装置18に存在することを検索する。

【００８２】ファイル共有インデックス管理機能部120は、時刻T32に検索結果を応答信号212としてファイル共有管理機能部160に出力する。ファイル共有管理機能部160は、時刻T34に応答信号216を操作インタフェース処理機能部162aに出力し、操作インタフェース処理機能部162aは、時刻T36に応答信号218を出力する。これにより、仮想的に構築した分散システムにおいてホスト端末装置16は、ファイル検索結果をこれまでよりも迅速に得ることができ、ディスプレイ168に表示させることができる。

【００８３】ファイル検索処理に続けて、たとえば、ファイル獲得を行う場合を説明する。この場合、ホスト端末装置16は、時刻T60で操作インタフェース処理機能部162aを介してファイル獲得の指示を入力する。操作インタフェース処理機能部162aは、時刻T62にファイル獲得要求をファイル共有管理機能部160に出力する。この要求を時刻T64で受けたファイル共有管理機能部160は、前述のファイル検索で得られたホスト端末装置18に対してファイル獲得要求として情報取得要求信号204を出力する。

【００８４】ここで、ファイル共有管理機能部160は、ファイル獲得要求をする前にローカルビューの視点でファイル検索結果が示すファイルパスからデータファイル1644内にこれまでにないディレクトリを含むか否かを調べる。ファイル共有管理機能部160は、ローカルビューになかったディレクトリがファイル検索結果に含まれている場合にファイル共有管理機能部180にこのディレクトリに対応するディレクトリ情報の獲得も要求する。

【００８５】ファイル共有管理機能部180は、時刻T68で受けた要求に応じて時刻T70にて対応するファイル情報、ファイルを取得情報信号206としてファイル共有管理機能部160に出力する。ファイル共有管理機能部160が上述したディレクトリ情報の獲得要求も合わせて出している場合、ファイル共有管理機能部180は、ディレクトリ情報も取得情報信号206に含めて送出する。

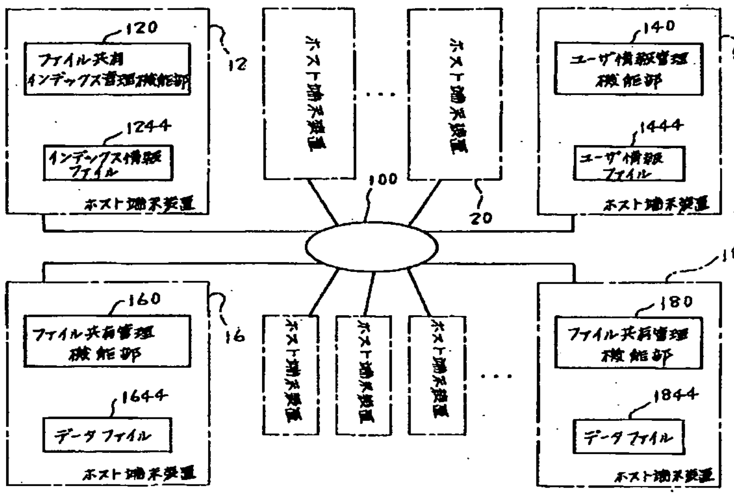
【００８６】ファイル共有管理機能部160は、供給された情報をキャッシュする（時刻T72）。そして、ファイル共有管理機能部160は、時刻T74にファイル共有インデックス管理機能部120に獲得した情報として情報通知信号212を通知する。ディレクトリ情報も獲得している場合、上述の情報にはディレクトリ情報も含むことは言うまでもない。ファイル共有インデックス管理機能部120は、排他制御を行いながら、時刻T76～T78の間にインデックス情報ファイル1244内にホスト端末装置16に新たな獲得した情報をインデックス情報として更新する。ファイル共有インデックス管理機能部120、ファイル管理機能部160、操作インタフェース処理機能部162aは、それぞれ、時刻T78，T80，T82にて更新完了を示す応答信号2

14, 216, 218を出力する。
【００８７】ファイル検索およびファイル獲得の処理において、処理を継続して行っていることから、アクセス権限の判定処理は１度しか行っていない。処理を継続せず、各処理を個々に操作する場合、各操作においてこの判定処理はそれぞれ１回ずつ行う。また、単なるファイル読出し操作の場合も分散ファイルシステム10では、前述したようにアクセス判定処理、アクセス制御処理、排他制御処理が行われることは言うまでもない。
【００８８】このようにそれぞれの場合に応じて仮想的に分散ファイルとして扱って、インデックス情報を基に集中管理して処理することにより、簡単に個々のアプリケーションレベルにおいてアクセス権限を判定して作成、更新、削除、検索、獲得を行う場合に比べて処理に要する所要時間を大幅に短縮することができる。
【００８９】以上のように構成することにより、分散ファイルシステム10は、ファイル共有インデックス管理機能部120に共有するファイルに関するインデックス情報を用いてインデックス情報からアクセス権限に応じた管理を行い、各ホスト端末装置16, 18, ･･･のうち、ユーザが操作するホスト端末装置が対象のディレクトリ情報を持っていなくても、各ホスト端末装置にアクセスすることなく、ファイル共有インデックス管理機能部120からアクセス権限を得ることができ、各ホスト端末装置16, 18, ･･･のファイル共有管理機能部160, 180, ･･･で局所的な管理を行うことにより、処理完了までに行う各ホスト端末装置16, 18, ･･･へのアクセスを最小限にすることができるので、処理の所要時間をこれまでよりも短時間で済ませることができる。したがって、分散ファイルシステム10は、使い勝手のよいシステムを提供することができる。
【００９０】また、ファイル共有インデックス管理機能部120が一元管理し、ホスト端末装置のそれぞれが有するディレクトリ情報を保持してアクセス権限の判定処理にこのディレクトリ情報を用いることにより、キャッシュしたファイルに対するアクセス権限の判定処理にともなってファイル共有インデックス管理機能部120へのアクセスをなくして、各ホスト端末装置が有するキャッシュファイルを有効に使用することができ、分散ファイルに対する処理を効率的に行わせることができる。
【００９１】特に、アクセス権限を変更した場合、変更したディレクトリをキャッシュしているホスト端末装置すべてに対してアクセス権限の更新が行われるので、システム中のアクセス権限の一貫性を保つことができる。
【００９２】
【発明の効果】このように本発明の分散ファイル共有システムおよびそのファイルアクセス制御方法によれば、大域管理機能ブロックに共有するファイルに関するインデックス情報を用いてインデックス情報からアクセス権限に応じた管理を行い、ユーザが操作するホスト端末装置が対象のディレクトリ情報を持っていなくても、各ホスト端末装置へのアクセスを行わずに、大域管理機能ブロックからアクセス権限を得ることができ、ファイル共有管理機能ブロックで局所的な管理を行うことにより、処理完了までに行う各ホスト端末装置へのアクセスを最小限にすることができるので、処理の所要時間をこれまでよりも短時間で済ませることができる。このシステムは、使い勝手のよいシステムとして提供することができる。
【００９３】また、大域管理機能ブロックが一元管理し、ホスト端末装置のそれぞれが有するディレクトリ情報を保持してアクセス権限の判定処理にこのディレクトリ情報を用いることにより、キャッシュしたファイルに対するアクセス権限の判定処理にともなって大域管理機能ブロックへのアクセスをなくして、各ホスト端末装置が有するキャッシュファイルを有効に使用することができ、分散ファイルに対する処理を効率的に行わせることができる。
【図面の簡単な説明】
【図１】本発明の分散ファイル共有システムを適用した分散ファイルシステムの概略的な構成を示す図である。
【図２】図１のホスト端末装置の内、ファイル共有インデックス管理を行うホスト端末装置の概略的な構成を示す図である。
【図３】図１のホスト端末装置の内、ユーザ情報管理を行うホスト端末装置の概略的な構成を示す図である。
【図４】図１のホスト端末装置の内、ファイル共有管理を行う一般的なホスト端末装置の概略的な構成を示す図である。
【図５】図１のホスト端末装置16におけるファイル管理を説明する図である。
【図６】図１のホスト端末装置18におけるファイル管理を説明する図である。
【図７】図１のホスト端末装置12におけるファイルの管理およびインデックス情報の管理を説明する図である。
【図８】図１のホスト端末装置の各ディレクトリが有するディレクトリ情報のデータ項目を説明する図である。
【図９】図８のディレクトリ情報の項目であるアクセス制御リストを説明する図である。
【図１０】図１のホスト端末装置の各ファイルが有するファイル情報のデータ項目を説明する図である。
【図１１】図５に示したホスト端末装置にてファイル獲得処理を行った後のファイル管理を説明する図である。
【図１２】図１の分散ファイルシステムにおけるアクセス権限の確認手順を説明するフローチャートである。
【図１３】図１の分散ファイルシステムにおける更新にともなうユーザ認証の手順を説明するフローチャートである。
【図１４】図１の分散ファイルシステムでのディレクトリおよびファイルに対する基本的な動作を示すシーケン

スチャートである。
【図１５】図１の分散ファイルシステムでのイベント通知に対する基本的な動作を示すシーケンスチャートである。
【図１６】図１の分散ファイルシステムでのファイル検索・ファイル獲得の動作を示すシーケンスチャートである。
【符号の説明】

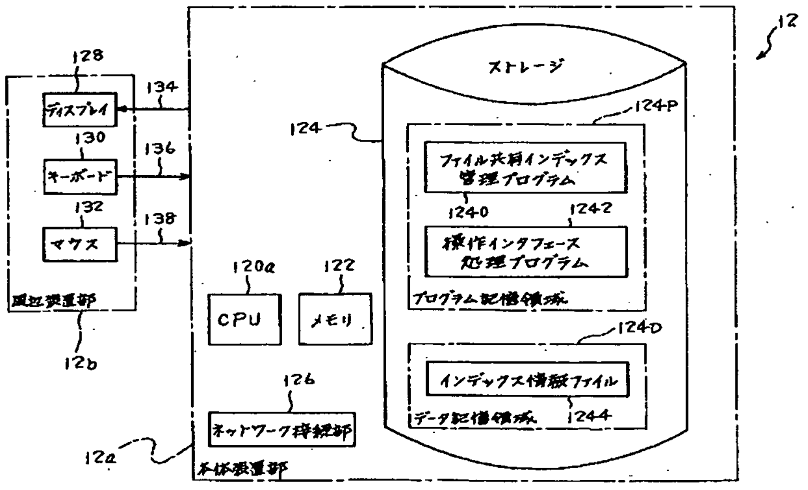
10	分散ファイルシステム
12〜20	ホスト端末装置
120	ファイル共有インデックス管理機能部
124, 144, 164, 184	ストレージ
140	ユーザ情報管理機能部
160, 180, 20a	ファイル共有管理機能部
1244	インデックス情報ファイル
1444	ユーザ情報ファイル
1644, 1844	データファイル

【図１】



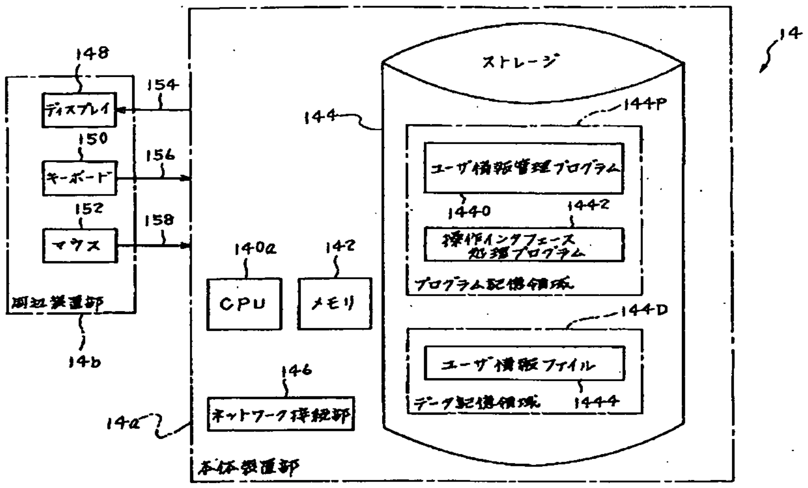
分散ファイル共有システムの構成例

【図２】



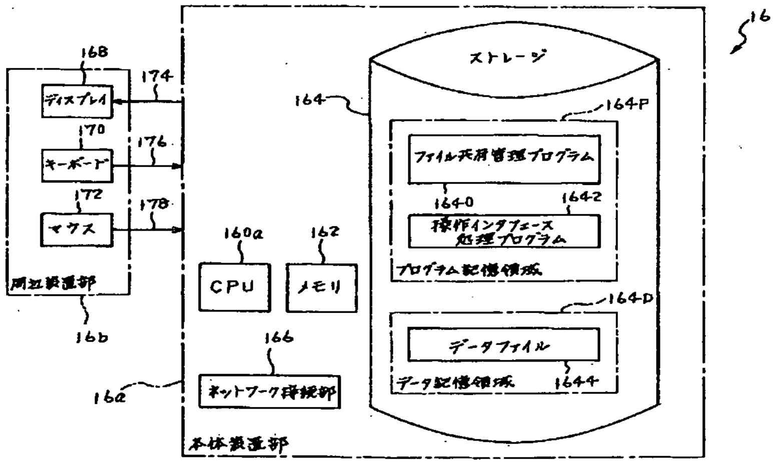
ホスト端末装置の構成例

【図3】



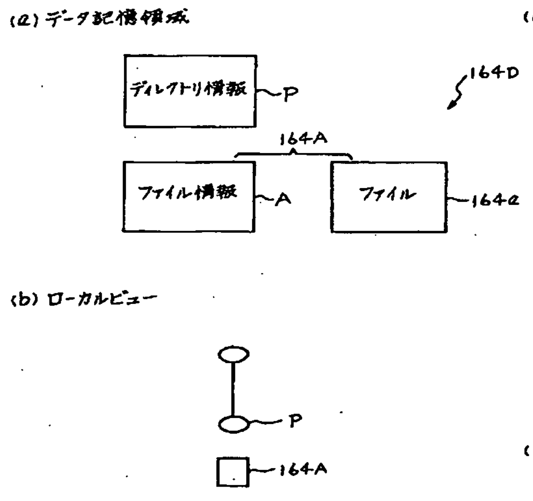
ホスト端末装置の構成例

【図4】



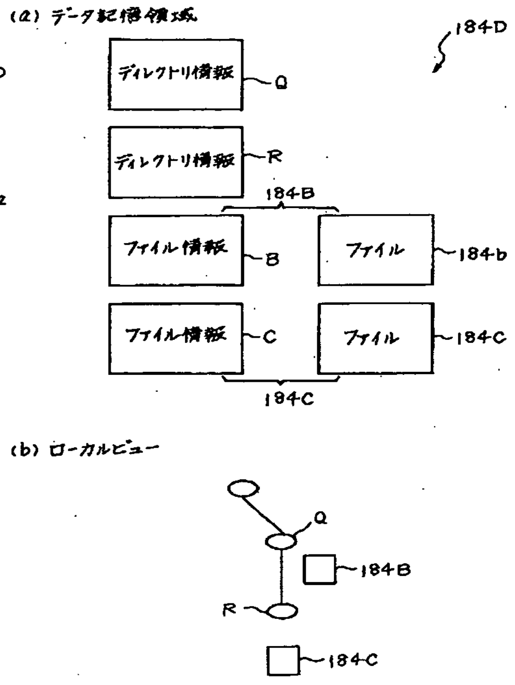
ホスト端末装置の構成例

【図5】



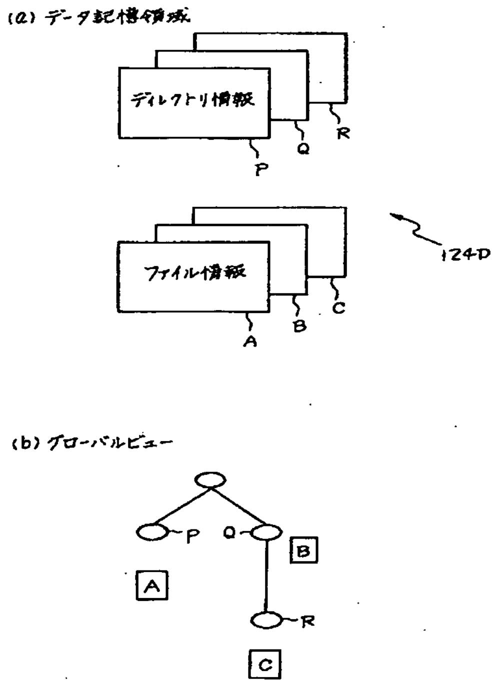
ホスト端末装置16のファイル管理

【図6】



ホスト端末装置18のファイル管理

【図7】



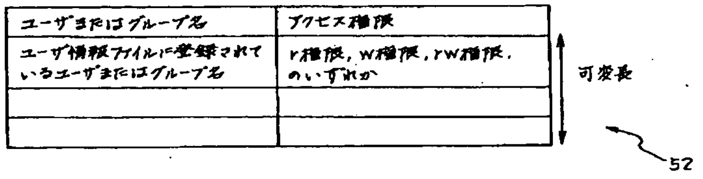
ホスト端末装置12のインデックス情報の管理

【図8】

キー	値	40
ディレクトリ名	本システムにおける仮想ディレクトリパス名	42
ディレクトリ更新日	ディレクトリが最後に更新された日	44
所有者	所有者名	46
破棄日	この日を過ぎると検索対象外となる。	48
公開日	この日以前は検索対象外となる。	50
アクセス権限	アクセス制御リスト	52

ディレクトリ情報

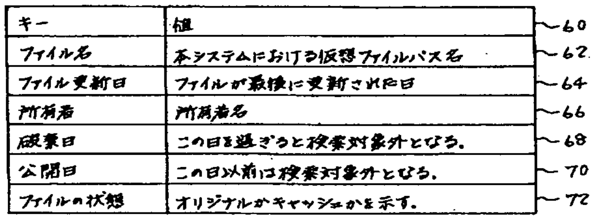
【図９】



ユーザまたはグループ名	アクセス権限
ユーザ情報ファイルに登録されているユーザまたはグループ名	r権限，w権限，rw権限，のいずれか

アクセス制御リスト

【図１０】

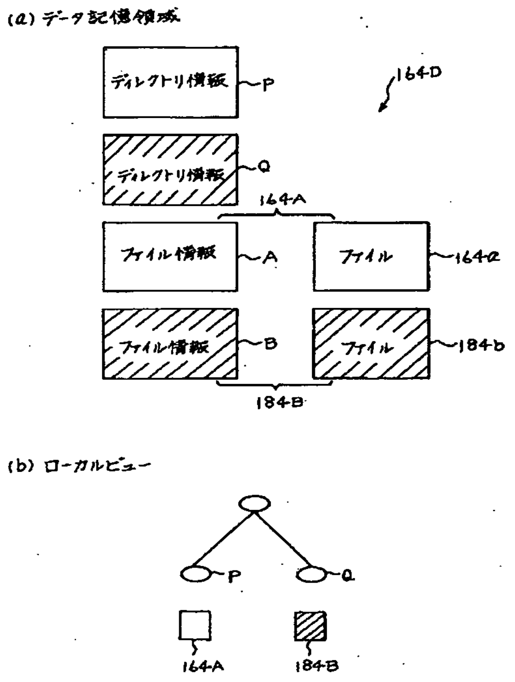


キー	値
ファイル名	本システムにおける仮想ファイルパス名
ファイル更新日	ファイルが最後に更新された日
所有者	所有者名
破棄日	この日を過ぎると検索対象外となる。
公開日	この日以前は検索対象外となる。
ファイルの状態	オリジナルかキャッシュかを示す。

60
62
64
66
68
70
72

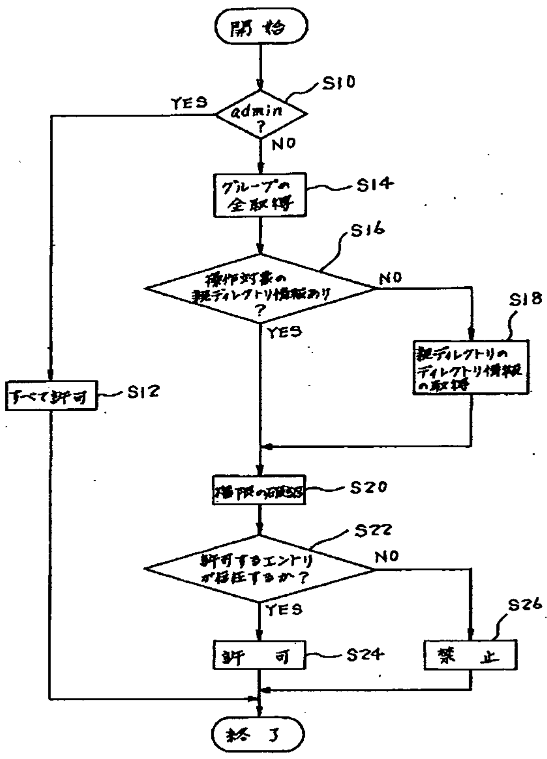
ファイル情報

【図１１】



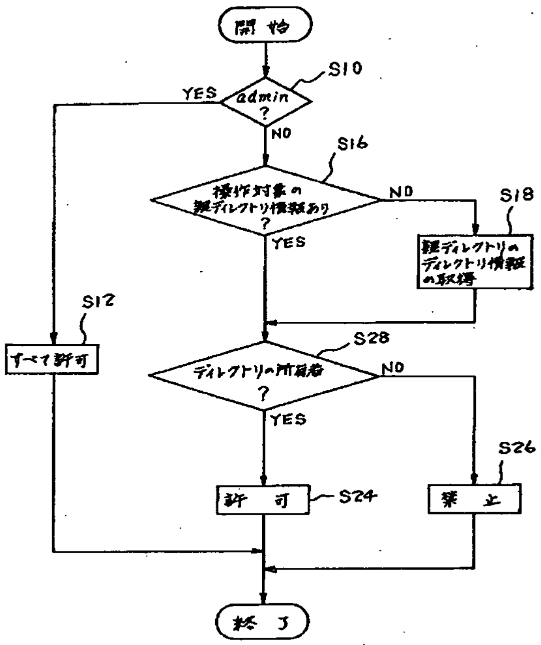
ファイル獲得処理後のファイル管理

【図１２】



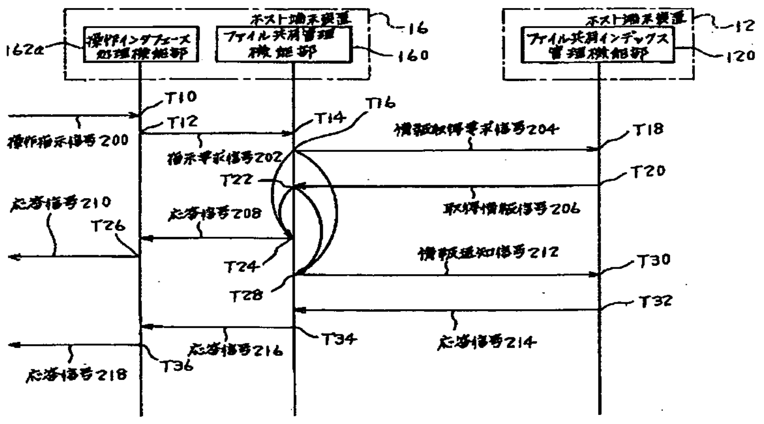
アクセス権限の確認手順

【図１３】



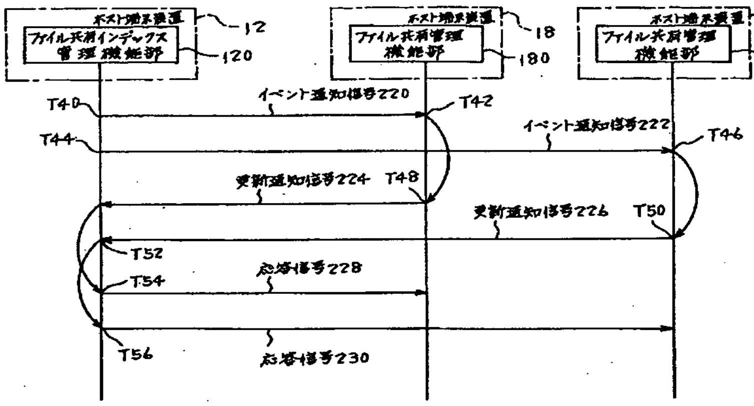
更新にともなうユーザ認証の手順

【図１４】



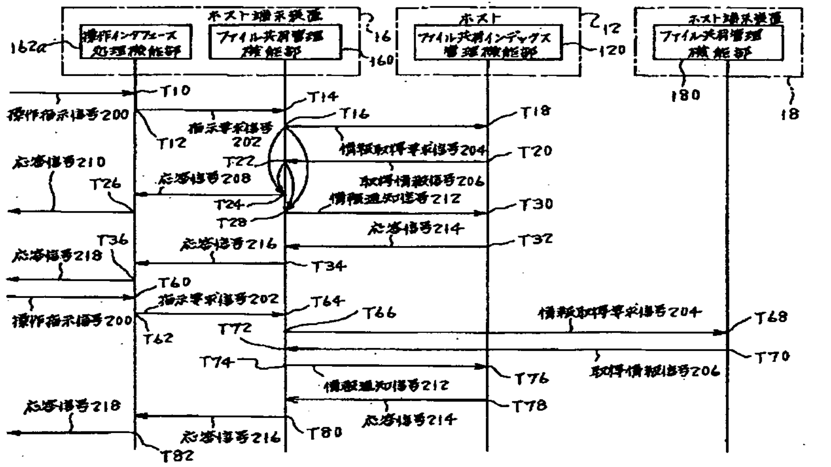
ディレクトリおよびファイルにおける基本的なシーケンスチャート

【図15】



イベント通知における基本的なシーケンスチャート

【図16】



ファイル検索・獲得シーケンスチャート

フロントページの続き

(51) Int. Cl.^{7}	識別記号	F I	テーマコード (参考)
G06F 17/30	120	G06F 17/30	120B
	150		150C

Fターム(参考)　5B075 KK03 KK43 KK54 KK63 KK64
KK66 KK67
5B082 EA11 FA16 GA13
5B085 AA01 AA08 AE04 BA07 BG03